NASA-CR-194857

IN-02-CR

202922

46P

Status Report for NASA Langley Grant NAG-1-1133: Development of a Code for Wall Contour Design in the Transonic Region of Axisymmetric and Square Nozzles

Authors:

Timothy Alcenius and Steven P. Schneider

School of Aeronautics and Astronautics

Purdue University

West Lafayette, IN 47907-1282

Principal Investigator:

Steven P. Schneider

Assistant Professor of Aerodynamics

School of Aeronautics and Astronautics

Purdue University

West Lafayette, IN 47907-1282

Period Covered: 9/1/92 to 1/1/94

Summary

Nozzle design codes developed earlier under NAG-1-1133 were modified and used in order to design a supersonic wind tunnel nozzle with square cross-sections. As part of the design process, a computer code was written to implement the Hopkins and Hill perturbation solution for the flow in the transonic region of axisymmetric nozzles. This technique is used to design the bleed slot of quiet-flow nozzles. This new design code is documented in this report.

Development of a Code for Wall Contour Design in the Transonic Region of Axisymmetric Nozzles

by

Timothy Alcenius (M.S.) and Steven P. Schneider (Assistant Professor)
School of Aeronautics and Astronautics
Purdue University
West Lafayette, IN 47907-1282

Abstract

An asymptotic analysis by D.F. Hopkins and D.E. Hill [1] for the flow in the transonic portion of an axisymmetric nozzle is used to develop a computer program for calculating the nozzle wall contour upstream of the nozzle throat. This code is to be integrated with Sivells [2] code for generating axisymmetric nozzles to create a program for calculating an entire nozzle contour. The development of the equations for the numerical solution is discussed along with a new technique for avoiding singularities. A test case was computed, and compared to results from Hopkins and Hill [1] and to results from the Douglas Aircraft implementation of Hopkins and Hill [3,4].

Introduction

Many methods are used to design two-dimensional axisymmetric nozzles, one being a characteristic method used in a program by Sivells [2]. This code, however, is limited when trying to resolve the flow field in the transonic portion of the nozzle. Another method was developed by D.F. Hopkins and D.E. Hill [1] in order to address this problem.

Originally, Hopkins and Hill's results were used by Douglas Aircraft Company to develop a computer code to calculate the flow in the transonic portion [4]. Even though it is still in use today, this code has two problems. First, the code is poorly documented and uses logic that is very hard to follow. It is therefore very difficult to understand and use. Second, it was programmed when computers were much less powerful than they are now.

Even though the solution appears accurate and would run quickly on any machine today, this made it even more difficult to understand.

The purpose of this paper is to describe a new computer program developed to implement the results of Hopkins and Hill. The paper includes the development of all the relations used by the code and a comparison of the results to the Douglas code and those given by Hopkins and Hill.

The nomenclature used here is identical to that described in Reference 1.

Development of Code

In order to use equations (17a-17d) as shown in Hopkins and Hill (also given in the appendix), H_r and M_r along with their derivatives with respect to ξ need to be calculated. Hopkins and Hill give a relation between H_r and ξ from one dimensional considerations along the axis in equation (22) of their paper (see appendix of this paper). From this, all of the derivatives were determined (see appendix). Note that θ_3' is required for the determination of M^*. Derivatives of $H_r(\xi)$ up to the fourth order are required for θ_3', along with M_r^* derivatives up to second order. These are shown below:

$$H_r' = \frac{\xi}{R_s}\exp\left\{-\frac{\xi^2}{2R_sC_1}\right\} \tag{1}$$

$$H_r'' = \left(\frac{1}{R_s} - \frac{\xi^2}{R_s^2C_1}\right)\exp\left\{-\frac{\xi^2}{2R_sC_1}\right\} \tag{2}$$

$$H_r''' = \left(\frac{\xi^3}{R_s^3C_1^2} - \frac{3\xi}{R_s^2C_1}\right)\exp\left\{-\frac{\xi^2}{2R_sC_1}\right\} \tag{3}$$

$$H_r'''' = \left(\frac{6\xi^2}{R_s^3C_1^2} - \frac{\xi^4}{R_s^4C_1^3} - \frac{3}{R_s^2C_1}\right)\exp\left\{-\frac{\xi^2}{2R_sC_1}\right\} \tag{4}$$

where R_s and C_1 are parameters that specify the shape of the reference boundary. Since the reference boundary and the wall of the nozzle will be of different shapes, R_s and C_1

flow field can be calculated for an arbitrarily defined nozzle shape. These functions are equations (33) and (34) given in [1] and are repeated in the appendix of this document.

Now that H_r and its derivatives are known, the equations for M_r and its derivatives can be obtained. The relations between M_r and H_r are given by equations (13) and (16) in Hopkins and Hill (also shown in the appendix). Using the chain rule with these equations, the first and second derivatives of M_r can be found. These are given below:

$$M_r' = \left(\frac{dM_r}{dM_r^*}\right)\left(\frac{dM_r^*}{dH_r}\right)H_r' \tag{5}$$

$$M_r'' = \left\{\left(\frac{d}{d\xi}\right)\left[\left(\frac{dM_r}{dM_r^*}\right)\left(\frac{dM_r^*}{dH_r}\right)\right]\right\}H_r' + \left(\frac{dM_r}{dM_r^*}\right)\left(\frac{dM_r^*}{dH_r}\right)H_r'' \tag{6}$$

The partial derivatives above are too long to be included here (they are included in the appendix). Knowing these variables, any values for ξ and η can be used to determine an x and y location, flow angle θ, and M^*.

As stated before, any value for η can be chosen to find a solution for equations (17a-17d) by Hopkins and Hill. However, the streamline that corresponds to the wall contour is the streamline of primary interest. Equations (17a-17d) in [1] have been normalized so that the value of y at the throat is 1.0. This means that the streamline that is desired is the one that passes through this point when the slope of the streamline is zero. This occurs when $dy/dx = 0$, or using the chain rule:

$$\frac{dy}{dx} = \left(\frac{dy}{d\xi}\right)\left(\frac{d\xi}{dx}\right) = 0. \tag{7}$$

Since $d\xi/dx$ is not zero in the field,

$$\frac{dy}{d\xi} = y_1'\eta + y_3'\eta^3 = 0 \tag{8}$$

is the equation of interest. This equation can be solved explicitly for η

$$\eta = \sqrt{-\frac{y_1'}{y_3'}} \quad . \tag{9}$$

with the values for y_1' and y_3' as given in the appendix.

The equation for η is then substituted into equation (17b) and y is set equal to 1.0. This equation is solved numerically using a bisection method. Then the value for the maximum streamline, η, is known from the solution to the above equation as well as the ξ location of the minimum point on the wall. This value of ξ is used to shift all of the x points so that x=0 will correspond to the minimum point on the wall instead of the location of the sonic point along the axis. Although this is not mentioned in Reference 1, this procedure is apparently used there also.

Finally, the initial and final value for ξ must be determined from the user input for θ_a and x_{end}. The variable θ_a represents the maximum angle that the wall makes upstream of the throat. The location of this angle is found by iterating over ξ using a bisection method to find where the calculated value of θ is equal to θ_a. Since direct computation of dy/dx did not agree with (17c) and was consistent with y(x), the calculated value for θ is found from the inverse tangent of dy/dx instead of by using equation (17c). Perhaps this mismatch indicates that an accurate description of θ requires more terms in the series. The equation that is used in the program is shown below:

$$\theta = \mathrm{TAN}^{-1}\left[\left(\frac{dy}{d\xi}\right)\left(\frac{d\xi}{dx}\right)\right] \tag{10}$$

where $dy/d\xi$ is the same as before and $d\xi/dx$ is given by:

$$\frac{d\xi}{dx} = \frac{1}{1 - x_2'\eta^2 - x_4'\eta^4} \, . \tag{11}$$

The values for x_2' and x_4' are given in the appendix. Using the value for η which corresponds to the wall, the value for ξ which makes θ equal to θ_a can be found.

The final value for ξ is determined from the input value for x_{end}. The variable x was chosen instead of θ for use as the ending variable so that a smooth transition to Sivells code [2] could be obtained. The value for ξ that corresponds to x_{end} at the wall for any other streamline is found using a bisection iteration scheme.

The equations must be solved at each point in the streamwise direction. Instead of using the streamwise coordinate ξ as the independent variable, M_r^* is used. This is so the iteration to find M_r^* only needs to occur to find the beginning and ending locations in the calculation. Given M_r^* at each location, all of the other variables can be solved for explicitly.

Difficulties were experienced due to singularities in M_r' and M_r'' at the sonic point. These were due to the fact that dM_r^*/dH_r goes to infinity and H_r' goes to zero at $M_r^*=1$ (see equation (5)). In order to avoid this problem, a perturbation solution was used in the vicinity of $M_r^*=1$. This solution gives:

$$\frac{dM_r^*}{dH_r}H_r' = \frac{dM_r^*}{d\xi} = \sqrt{\frac{2}{(\gamma+1)R_s}} \tag{12}$$

where R_s is the same as before and is shown in the appendix. This equation yields results that are smooth and correct in the region of $M_r^* = 1$. A perturbation solution was also used for M_r''. This was derived by rewriting the equation as:

$$M_r'' = \left[\left(\frac{d}{d\xi}\right)\left(\frac{dM_r}{dM_r^*}\right)\right]\left(\frac{dM_r^*}{d\xi}\right) + \left[\left(\frac{d}{d\xi}\right)\left(\frac{dM_r^*}{d\xi}\right)\right]\left(\frac{dM_r}{dM_r^*}\right). \tag{13}$$

Since in the area of $M_r^* = 1$, $dM_r^*/d\xi$ is constant, it is assumed that the derivative with respect to ξ of this function is zero. This leaves:

$$M_r'' = \left[\left(\frac{d}{d\xi}\right)\left(\frac{dM_r}{dM_r^*}\right)\right]\left(\frac{dM_r^*}{d\xi}\right) \qquad (14)$$

which yields a continuous variation of M_r'' in this region.

The final decision that had to be made was the region in which to apply these solutions. As can be seen in figure #1, Mr* = 1±.01 gives reasonable results.

Results

The validity of the results were checked in two different ways. First, the plots that are given by Hopkins and Hill were reproduced. These can be seen in figures 2 to 4. The data appear correct, but a more precise comparison was desirable.

In order to make a second check, results from the Douglas program [3,4] were used. Given the variables shown below, the streamline that corresponds to the wall contour were compared.

Table #1 - Parameters Used in Comparison with Langley Results

R_a	1.0
θ_a	45°
γ	1.4

The results are shown in figures 5 to 8. As can be seen in these figures, the results from the new code appear to match precisely with the results given by the Douglas code [3,4]. By checking the numerical results from both methods, it was found that the codes always agree to at least four figures for all variables except for θ which agrees to two figures. Figure 8 also shows that the solution given by the new program is more continuous near M* = 1 than the Douglas code [3,4]. Therefore, the new program yields a slightly more accurate solution than was previously available.

Conclusions

A new FORTRAN-77 computer code has successfully been developed for calculating the transonic flow field in an axisymmetric nozzle. The results of this code are in excellent agreement with results from a previous FORTRAN-4 code. The code is well documented and commented to allow ready use. The algorithm for the code is also provided so that the user may have a better understanding of the equations and the logic behind the code development.

This code has also been integrated with the Sivells [2] code. The integrated program is capable of generating an entire nozzle contour, including a region upstream of the throat. Although Sivells uses Hall's transonic solution downstream of the throat [2], the two asymptotic solutions should agree very near the throat. Current results, to be reported on elsewhere, indicate that this is the case.

References

1. Hopkins, D.F. and Hill, D.E. *Effects of Small Radius of Curvature on Transonic Flow in Axisymmetric Nozzles*, AIAA Journal Vol.4 No. 8, August 1966.

2. Sivells, John. *A Computer Program for the Aerodynamic Design of Axisymmetric and Planar Nozzles for Supersonic and Hypersonic Wind Tunnels*, AEDC-TR-78-63, December 1978.

3. Chen, Frank. NASA Langley Research Center, Private Communication, May 1993.

4. Rustoi, Hill, Hopkins. FORTRAN-4 Program *F479*, Implementation of Hopkins and Hill by the Douglas Aircraft Company.

Appendix

The following equations were taken directly from Hopkins and Hill (1). They are placed here fore the readers convenience.

$$H_r^2 = \frac{1}{M_r^*}\left[\frac{\gamma+1}{2} - \frac{\gamma-1}{2}M_r^{*2}\right]^{-\left(\frac{1}{\gamma-1}\right)} \tag{13}$$

$$M_r^2 = \frac{2M_r^{*2}}{\left[\gamma+1-(\gamma-1)M_r^{*2}\right]} \tag{16}$$

$$x = \xi - \frac{H_r H_r'}{2}\eta^2 - \left[\frac{3H_r^2 H_r' H_r''(M_r^2-1)}{32} - \frac{H_r(H_r')^3}{96} + \theta_3\frac{H_r}{4}\right]\eta^4 \tag{17a}$$

$$y = H_r\eta + \frac{H_r}{8}\left[H_r H_r''(M_r^2-1) - (H_r')^2\right]\eta^3 \tag{17b}$$

$$\begin{aligned}\theta = H_r'\eta + \Bigg[&\frac{H_r H_r' H_r''}{6}(2M_r^2-1) + \frac{H_r^2 H_r'''}{8}(M_r^2-1) + \frac{H_r^2 H_r''}{4}M_r M_r' - \\ &\frac{H_r H_r' H_r''}{12}(M_r^2-2) + \frac{(H_r')^3}{24}\Bigg]\eta^3\end{aligned} \tag{17c}$$

$$\frac{M^*}{M_r^*} = 1 + \frac{H_r H_r''}{2}\eta^2 + \left[\frac{H_r^2(H_r'')^2}{4} + \frac{3H_r^2(H_r'')^2}{32}(M_r^2-1) + \frac{H_r H_r''(H_r')^2}{32} + \frac{H_r\theta_3'}{4}\right]\eta^4 \tag{17d}$$

$$H_r = C_1\left[1 - e^{\left(-\frac{\xi^2}{2R_a C_1}\right)}\right] + 1 \tag{22}$$

$$\begin{aligned}R_s = (0.9322 + 0.0565\gamma)\Big[&0.6173e^{\left(0.48C_1^{-0.189}\right)} + R_a(1.1234 + 0.00771C_1 - \\ &0.000163C_1^2) - R_a^2(0.0182 + 0.00111C_1 - 0.0000201C_1^2)\Big]\end{aligned} \tag{33}$$

$$C_1 = \left[\frac{\arctan\left[\dfrac{\ln \theta_a}{\ln 15000 - \ln R_a} \right]}{19.55 + 1.25(\gamma - 1.4)} \right]^{\frac{1}{[0.1277 - 0.0355(\gamma - 1.4)]}} \tag{34}$$

Here, R_a and θ_a are the actual values for the radius of curvature and maximum inflow angle that are input by the user.

The following equations were derived for use in the program.

$$\theta_3' = \frac{1}{4}\{H_r'''(H_r H_r' M_r^2 + H_r^2 M_r M_r') + H_r''[(H_r')^2 M_r^2 + H_r H_r'' M_r^2 + 4H_r H_r' M_r M_r' + H_r^2 (M_r')^2 + H_r^2 M_r M_r'']\} + \frac{1}{8}\{(2H_r H_r' H_r''' + H_r^2 H_r'''')(M_r^2 - 1) + 2H_r^2 H_r''' M_r M_r' + (H_r')^2 H_r''\}$$

$$\frac{d}{d\xi}\left[\left(\frac{dM_r}{dM_r^*}\right)\left(\frac{dM_r^*}{dH_r}\right)\right] = \frac{S_1' S_2 (S_3 - S_4) - S_1[S_2'(S_3 - S_4) + S_2(S_3' - S_4')]}{S_2^2(S_3^2 - S_4^2)}$$

where

$$S_1 = (\gamma + 1)M_r$$
$$S_2 = M_r^{*2}$$
$$S_3 = H_r$$
$$S_4 = \frac{H_r}{M_r^2}$$
$$S_1' = (\gamma + 1)M_r'$$
$$S_2' = 2M_r^* M_r^{*\prime}$$
$$S_3' = H_r'$$
$$S_4' = \frac{H_r' M_r^2 - 2M_r M_r' H_r}{M_r^4}$$
$$M_r^{*\prime} = \frac{M_r'}{\left(\dfrac{dM_r}{dM_r^*}\right)}$$
$$\frac{dM_r}{dM_r^*} = \frac{2M_r^*(\gamma + 1)}{M_r[\gamma + 1 - (\gamma - 1)M_r^{*2}]^2}$$

$$y_1' = H_r'$$

$$y_3' = \frac{1}{8}\left\{H_r'\left[H_r H_r''\left(M_r^2 - 1\right) - \left(H_r'\right)^2\right] + H_r\left[H_r' H_r''\left(M_r^2 - 1\right) + H_r H_r'''\left(M_r^2 - 1\right) + 2H_r H_r'' M_r M_r' - 2H_r' H_r''\right]\right\}$$

$$x_2' = -\left[\frac{\left(H_r'\right)^2}{2} + \frac{H_r H_r''}{2}\right]$$

$$x_4' = -\frac{1}{96}\left\{9\left[2H_r\left(H_r'\right)^2 H_r''\left(M_r^2 - 1\right) + H_r^2\left(H_r''\right)^2\left(M_r^2 - 1\right) + H_r^2 H_r' H_r'''\left(M_r^2 - 1\right) + 2H_r^2 H_r' H_r'' M_r M_r'\right] - \left[\left(H_r'\right)^4 + 3H_r\left(H_r'\right)^2 H_r''\right] + 24\left(H_r \theta_3' + H_r' \theta_3\right)\right\}$$

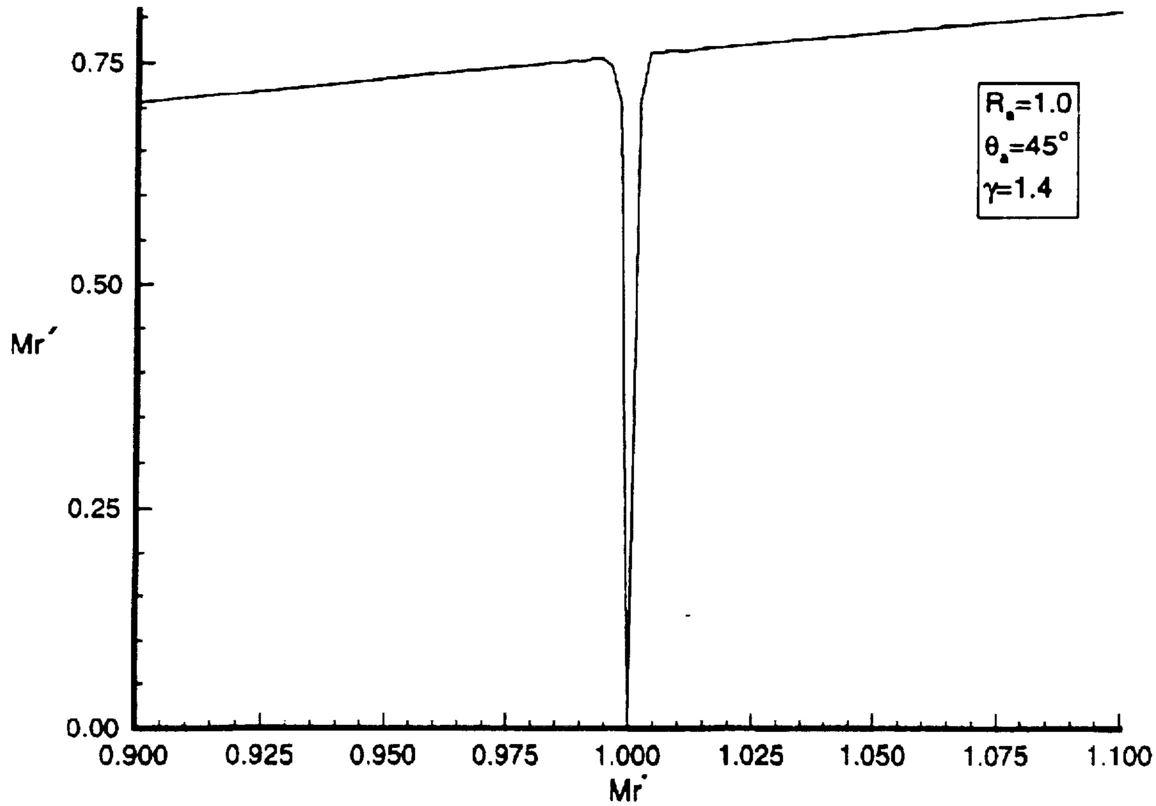

Figure #1-Region of Discontinuity in Mr'

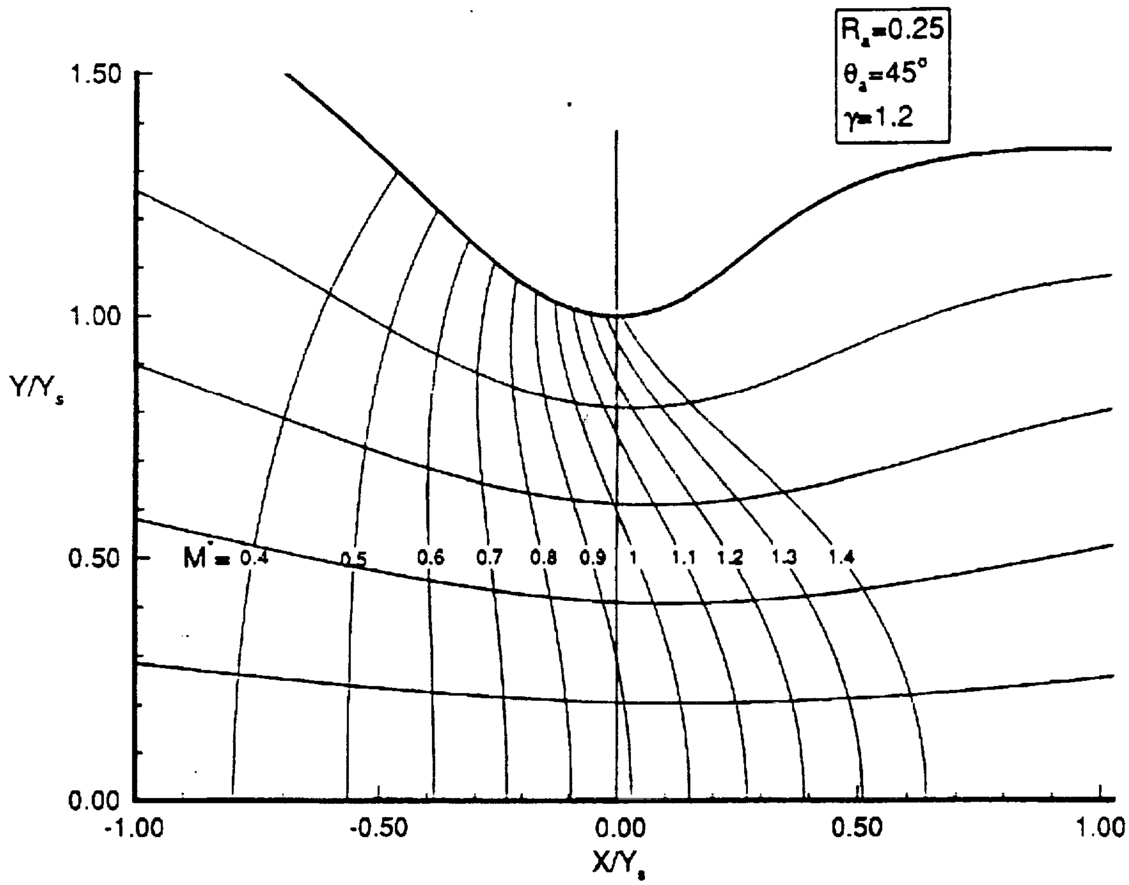

Figure #2-Re-computation of Figure 4 from Hopkins and Hill [1]

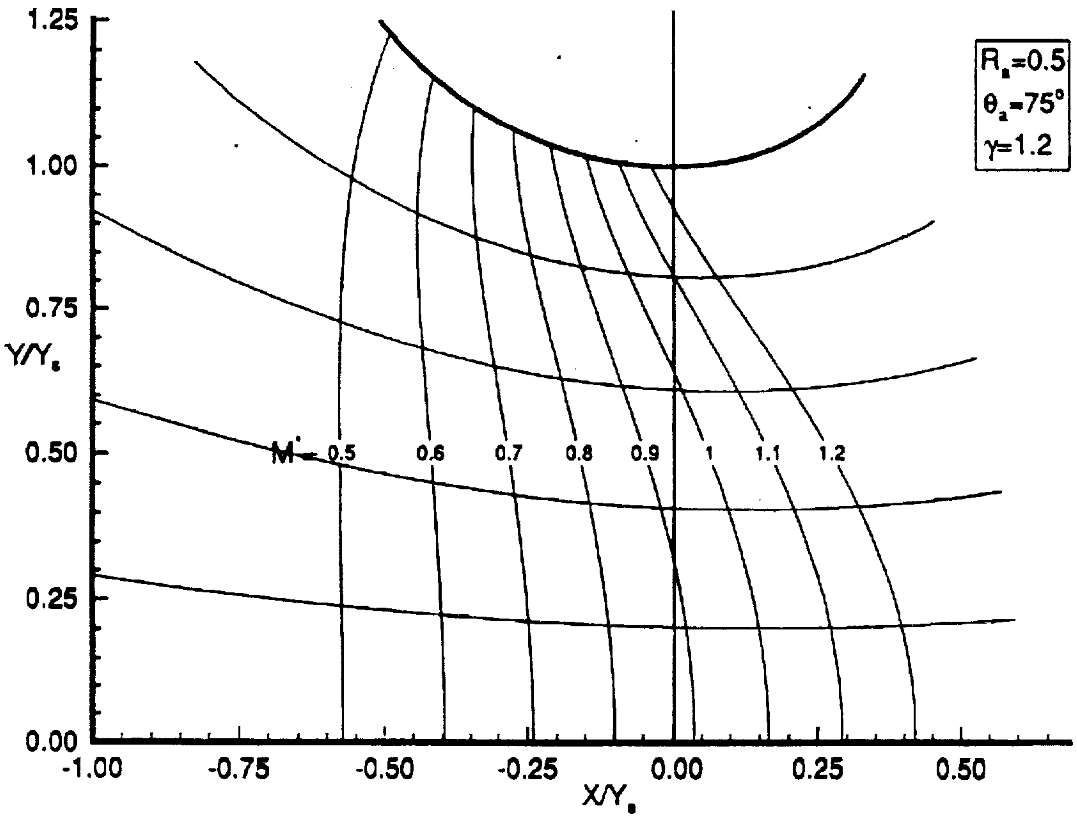

Figure #3-Re-computation of Figure 5 from Hopkins and Hill [1]

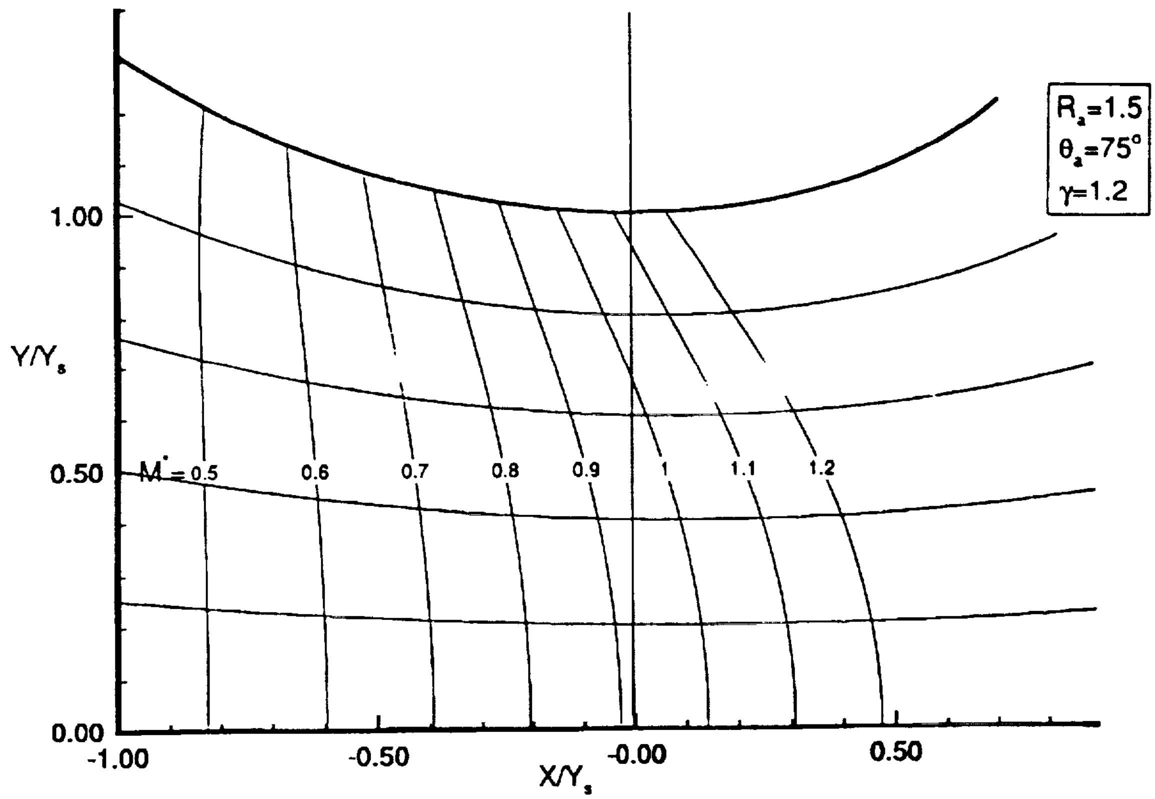

Figure #4-Re-computation of Figure 6 from Hopkins and Hill [1]

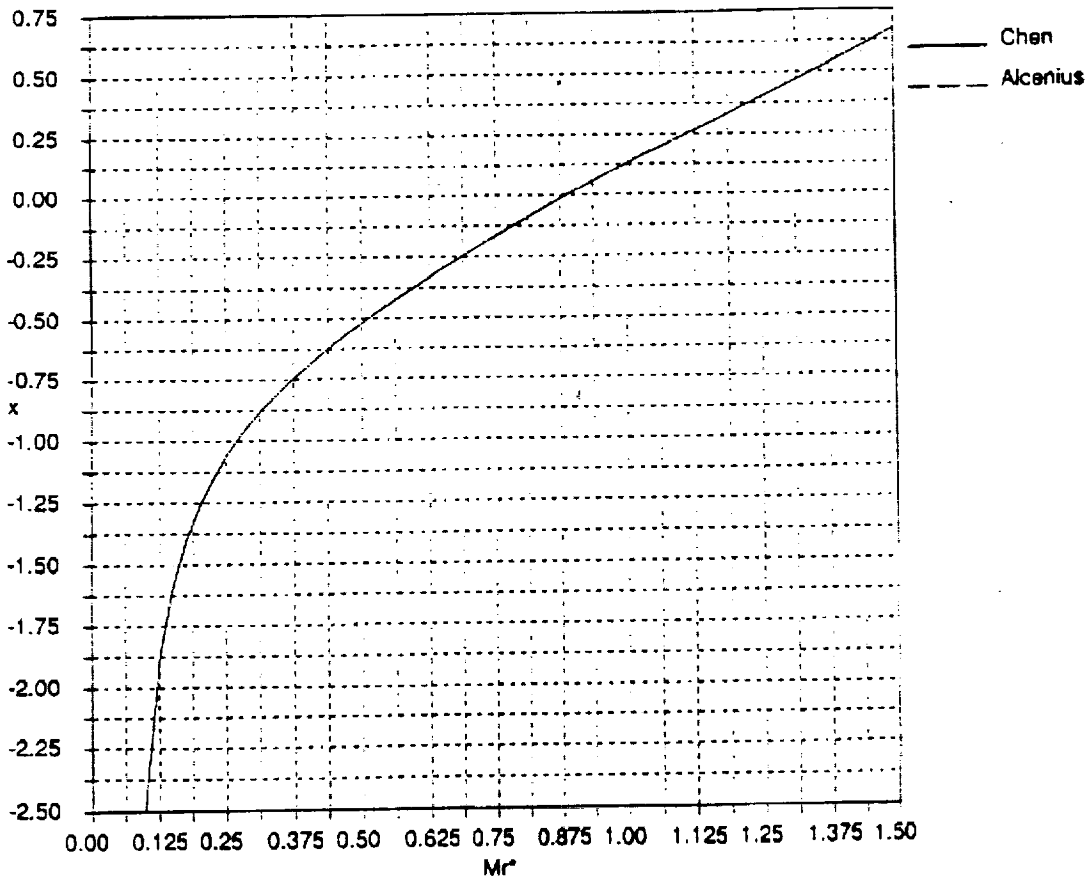

Figure #5-Comparison of X between Current Program and Douglas Program

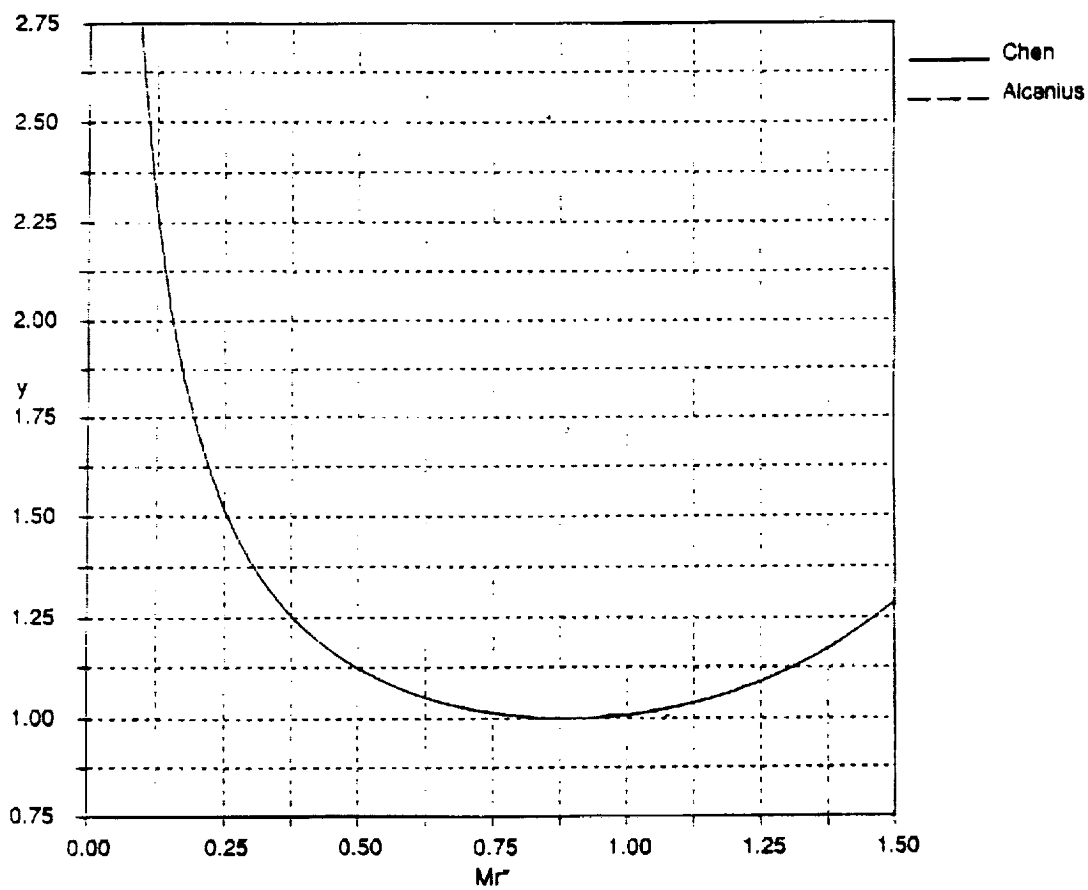

Figure #6-Comparison of Y between Current Program and Douglas Program

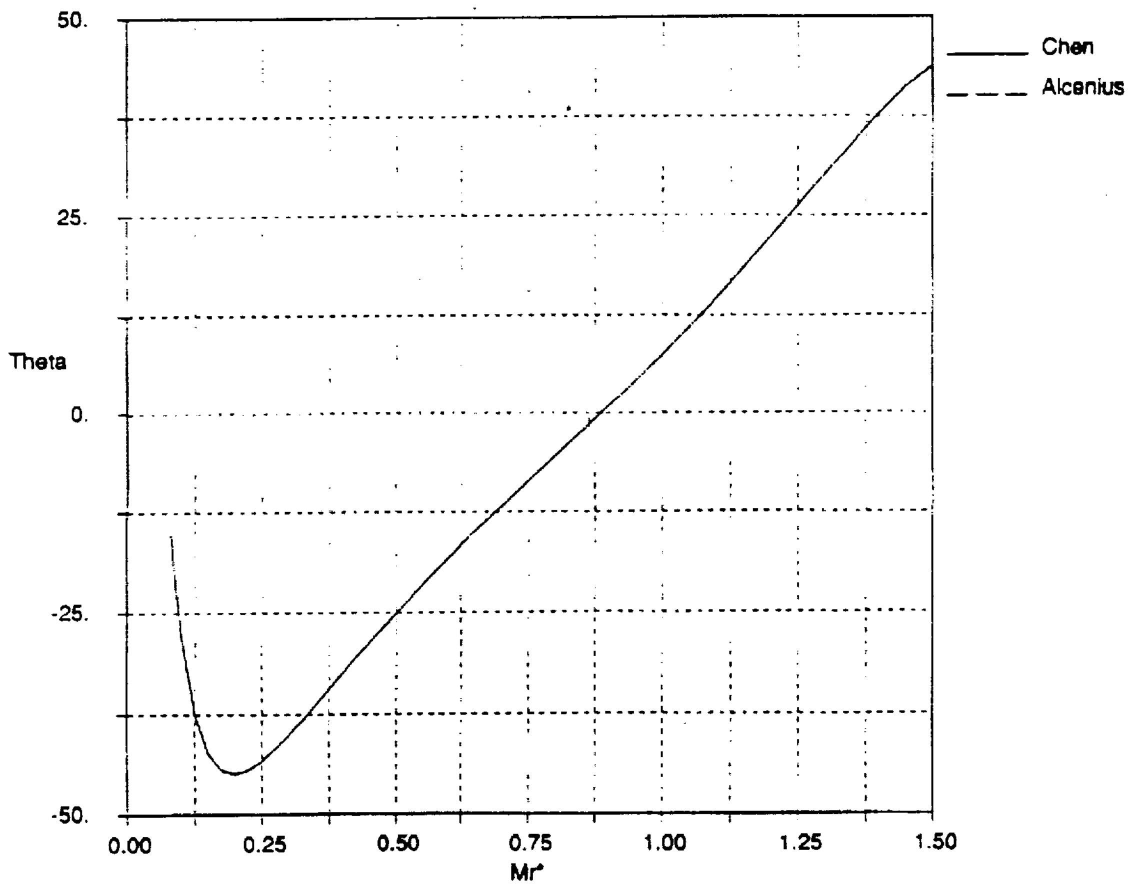

Figure #7-Comparison of θ between Current Program and Douglas Program

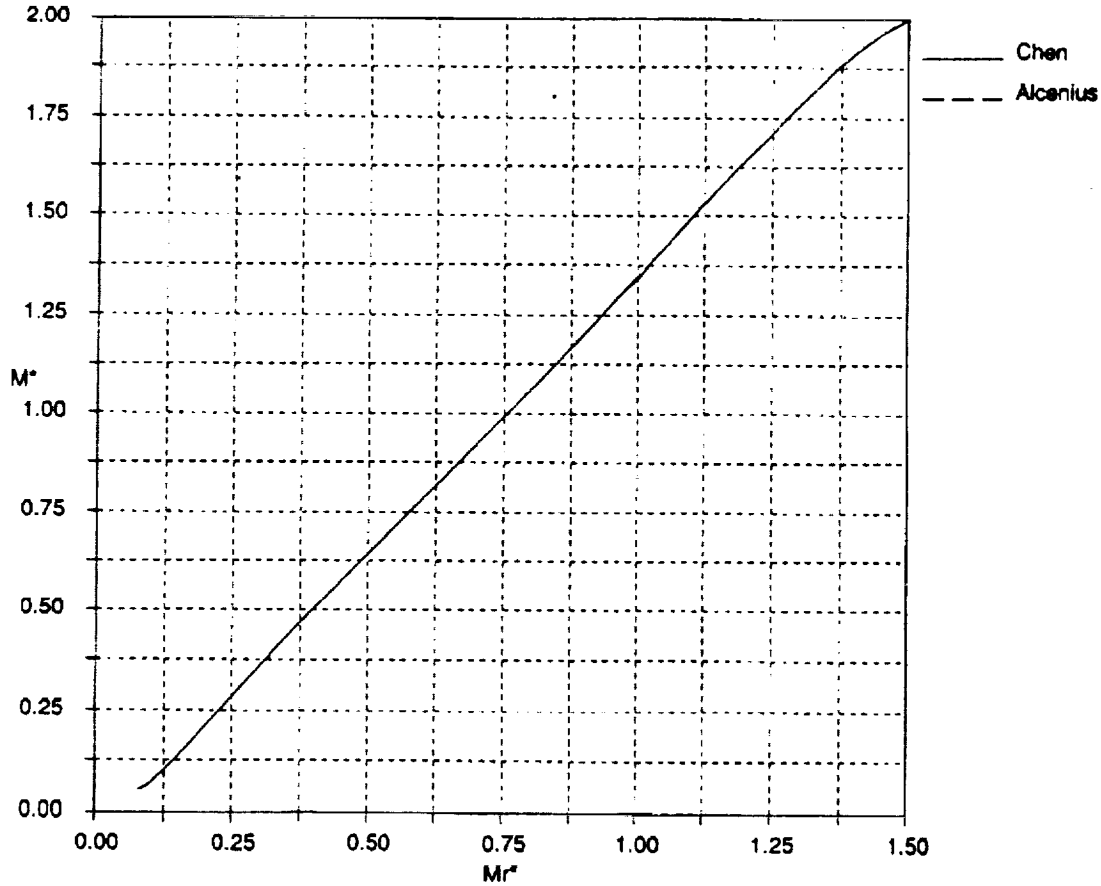

Figure #8-Comparison of M* between Current Program and Douglas Program

Appendix

An electronic version of the source for the Fortran-77 code can be obtained from the Experimental Methods Branch at the NASA Langley Research Center, Hampton, VA, 23681. Contact Dr. Stephen Wilkinson. Alternatively, contact Prof. Steven P. Schneider, Aeronautical and Astronautical Engineering, Purdue University, West Lafayette, IN 47907-1282, email `steves@ecn.purdue.edu`, telephone (317) 494-3343.

The following pages contain a printed version of the current source code.

```
*  This is a program to test the hopkins-hill subroutine sps 6-2-93
*     subroutine hophill(ra,thetaa,gam,thetae,yt,n,xs,ys,amachs)
      real xs(100),ys(100),amachs(100)
      data n/100/
*
      print *, 'What is the radius of curvature?'
      read *, ra
      print *, 'What is the initial angle (degrees)?'
      read *, thetaa
      print *, 'What is the ratio of specific heats?'
      read *, gam
      print *, 'What is the final value of x?'
      read *, xexit
*
      write (*,*) 'opening hophill.tst for test output'
      open (unit=1,file='hophill.tst')
      write (1,*) 'x,y,amach along contour'
      yt = 1.0
      call hophill(ra,thetaa,gam,xexit,yt,n,xs,ys,amachs)
      write (*,*) 'returned from first call to hophill'
      do 50 i = 1,n
        write (1,*) xs(i),ys(i),amachs(i)
   50 continue
      yt = 0.95
      write (1,*) 'x,y,amach along streamline through yt= ',yt
      call hophill(ra,thetaa,gam,xexit,yt,n,xs,ys,amachs)
      do 80 i = 1,n
        write (1,*) xs(i),ys(i),amachs(i)
   80 continue
      stop 'normal end'
      end
```

```
* From alcenius Wed Jun  2 02:58:29 1993
*     This program is developed to calculate the transonic flow
*     region of an axisymmetric nozzle.  The equations are taken from
*     Hopkins, D.F. and Hill, D.E., "Effect of Small Radius of
*     Curvature on Transonic Flow in Axisymmetric Nozzles", AIAA
*     Journal Vol.4, PP. 1337-1343 (August 1966).
*     Written by Tim Alcenius 5-24-93.
*
*     The variables in the main program are defined as follows:
*
*     ra - The actual radius of curvature input by the user
*     thetaa - The actual initial flow angle of the wall input by
*              the user.
*     xend - The last value of x that data is written out for,
*              input by the user.
*     gam - The ratio of specific heats input by the user.
*     c1 - Reference bounday shape parameter for flow angle Eqn. 34.
*     rs - Reference bounday shape parameter for radius of curvature
*          Eqn. 33.
*     c1top - The numerator of Eqn #34
*     c1bottom - The denominator of Eqn #34
*     gmp - Gamma + 1
*     gmm - Gamma - 1
*
*     The first part of the program defines all of the variables and
*     reads in the user directed input for the problem.
* This is a subroutine version of the same program, modified to
*  allow finding interior streamlines.  sps 6-2-93
*
      subroutine hophill(ra,thetaa1,gam1,xend1,yt1,n,xs,ys,amachs)
      parameter (ndim=2)
* xs,ys are arrays with coords of line, amachs is mach no. at pt, n is no. pts.
* yt is the throat value of y for the streamline to be returned
      real ra, thetaa, gam, gmm, gmp, c1top, c1bottom, c1, rs, mst
      real xend, pi, mrst, mrstthroat, mrstbeg,mrstend,mrstnow,mrststr
      real xs(n),ys(n),amachs(n)
      real stpts(ndim+1,ndim) !initial simplex guess for AMOEBA
      real stans(ndim+1) !values of GETSTR at three pts in stpts
      real guess(ndim)  !for guesses
      logical linit,lsuper1  !initialization variable, first call, yt=1
      save linit
      external amoeba,getstr,getxy
      common/stream/yt  !to pass the desired value of yt to getstr
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/throat/etaw,xithroat,mrstthroat,xsl0 !results at wall,throat from
* first iteration for continuation. Note that x,y = 0,1 at throat!
* remember that mrst depends only on xi, not on eta!
      common/ends/mrstbeg,mrstend,thetaa,xend,thetanow,mrstnow,xbeg
*       !mrst at entry and exit, set in MRSTBAE, theta at entry and exit
      data tiny/1.0e-5/  !allowable error in iteration
      data ftol/1.0e-6/  !allowable error in AMOEBA iteration
*
      thetaa = thetaa1
      xend = xend1
      yt = yt1
      gam = gam1  !to avoid fortran error, formal argument illegal in common
      if (yt .eq. 1.0) then  !initialize
        if (linit) then
          write (*,*) 'should only call with yt=1 the first time!'
          stop 'fatal error'
```

```
        else
          linit = .true.
        end if
        pi = ACOS(-1.)
*       write (*,*) 'What is the radius of curvature?'
*       read *, ra
*       write (*,*) 'What is the initial angle (degrees)?'
*       read *, thetaa
*       write (*,*) 'What is the ratio of specific heats?'
*       read *, gam
        gmm = gam - 1.
        gmp = gam + 1.

*     This part of the program calculates the top and bottom portions
*     of eqn #34 and the calculates the reference boundary shape
*     parameters using Eqn #33 and #34 from the paper.

        c1top = 180.*ATAN((LOG(thetaa)/(LOG(15000.0)-LOG(ra))))/pi
        c1bottom = 19.55 + 1.25*(gam - 1.4)
        c1 = (c1top/c1bottom)**(1./(0.1277 - (gam-1.4)*0.0335))
        rs = (0.9322 + 0.0565*gam)*(0.6173*EXP(0.48/c1**(0.189)) +
     +    ra*(1.1234 + 0.00771*c1 - 0.000163*c1**2) - ra**2*(0.0182 +
     +    0.00111*c1 - 0.0000201*c1**2))

*     This part sets the angles to radians and calls the program to
*     calculate the flow field properties in the throat area.

        thetaa = -thetaa*pi/180  !here correct signs so input>0
        call mstar  !sets up mrstbeg and mrstend
*       write (*,*) 'HOPHILL: calling getxy with throat conditions'
*       write (*,*) 'mrstthroat= ',mrstthroat,' etaw= ',etaw
        call getxy(mrstthroat,etaw,xi,x,y,mst) !returns xi, x, y, and mst
*                   for guess of mrst and eta
        if (abs(xi-xithroat) .gt. tiny) then
          write (*,*) 'HOPHILL: xi error in getxy'
        end if
        if (abs(x).gt. tiny) write (*,*) 'x error in getxy'
        if (abs(y-1.0).gt.tiny) write (*,*) 'y error in getxy'
        write (*,*) 'HOPHILL: proceeding to compute contour pts'
* Now have the setup for the streamline through the y=1 pt.  Now
*  get the coords along the streamline.  use the subroutine
        delmrst = (mrstend-mrstbeg)/float(n-1)
        do 100 i = 1,n
          mrst = mrstbeg+delmrst*(i-1)
          call getxy(mrst,etaw,xi,x,y,mst) !returns xi,x,y,mst for mrst and eta
          xs(i) = x
          ys(i) = y
          amach = sqrt( 2.0*mst**2/( gmp - gmm*mst**2) )  !eqn 16
          amachs(i) = amach
  100   continue
        return  !from first call
      else  !call for streamline, use same xbeg and xend as before!
* Note that lines of constant eta are streamlines - see eqn (9) in paper
        if (.not. linit) then
          write (*,*) 'should call with yt=1 the first time!!'
          stop 'fatal error'
        end if
* Here, iterate for the values of eta and xi at the throat
*    for the streamline selected
* Do this using minimization of (x**2 + (y-yt)**2), Numerical Recipes
```

```
*   routine amoeba, see also Hildebrand.  Function GETSTR implements part.
* Need good guesses.  Note that eta approx. = y near throat.  Basis
*   of second guess.
        stpts(1,1) = mrstthroat
        stpts(1,2) = etaw  !throat is first pt, space is (mrst,eta)
        stpts(2,1) = 1.0  !sonic pt near x=0,y=yt
        stpts(2,2) = yt
        stpts(3,1) = 0.95  !just upstream
        stpts(3,2) = yt
        do 280 i = 1,ndim+1
          guess(1) = stpts(i,1)
          guess(2) = stpts(i,2)
          stans(i) = getstr(guess)
  280   continue
        call amoeba(stpts,stans,ndim+1,ndim,ndim,ftol,getstr,iter)
        write (*,*) 'HOPHILL: AMOEBA returns after ',iter,' iterations'
        write (*,*) 'values at three points are:  (yt= ',yt,')'
        write (*,*) '(mrst,eta,x,y,abs(x)+abs(y-yt),xi)'
        toterror = 0.0
        do 290 i = 1,ndim+1
          call getxy(stpts(i,1),stpts(i,2),xi,x,y,mst)
*         guess(1) = stpts(i,1)
*         guess(2) = stpts(i,2)
*         zeroit = getstr(guess)
*         write (*,*) 'HOPHILL: guess,zeroit= ',guess,zeroit
          write (*,288) stpts(i,1),stpts(i,2),x,y,stans(i),xi
  288     format(1x,6(f12.7,1x))
          toterror = toterror+stans(i)
  290   continue
        if (toterror .gt. tiny) then
          write (*,*) 'HOPHILL: toterror is ',toterror
          write (*,*) 'converged to poor solution from AMOEBA'
          stop 'not worth continuing'
        end if
        mrststr = (stpts(1,1)+stpts(2,1)+stpts(3,1))/3.0
        etastr = (stpts(1,2)+stpts(2,2)+stpts(3,2))/3.0
* Now get New mrstbeg and mrstend for this new streamline, using xbeg and xend
        lsuper1 = .false.
        mrstbeg = findmrst(etastr,xbeg,lsuper1)  !function for this purpose
        lsuper1 = .true.  !mrst supersonic at throat or downstream (Not mst!)
        mrstend = findmrst(etastr,xend,lsuper1)
*
        delmrst = (mrstend-mrstbeg)/float(n-1)
        write (*,*) 'mrststr,etastr,delmrst= ',mrststr,etastr,delmrst
        do 300 i = 1,n
          mrst = mrstbeg+delmrst*(i-1)
          call getxy(mrst,etastr,xi,x,y,mst) !returns xi,x,y,mst for mrst,etastr
          xs(i) = x
          ys(i) = y
          amach = sqrt( 2.0*mst**2/( gmp - gmm*mst**2) )  !eqn 16
          amachs(i) = amach
  300   continue
        return
      end if
      end

* FUNCTION GETSTR
* this function serves to interface between subroutine GETXY and
* the Numerical recipes routine AMOEBA
*
```

```
      function getstr(guess)
      real guess(2),mrst,mst  !(mrst and etast)
      common/stream/yt !desired streamline is through x=0,y=yt
*
      mrst = guess(1)
      etast = guess(2)
      call getxy(mrst,etast,xi,x,y,mst)
      getstr = abs(x) + abs(y-yt)  !minimize this, will have soln
*     write (*,1) x,y,getstr,yt
*   1 format(' GETSTR: x,y,getstr,yt= ',4(f13.7,1x))
*     write (*,*) 'mrst,etast= ',mrst,etast
      return
      end

* FUNCTION GETX2
      function getx2(mrst)
      real mrst,mstout
* given mrst, passed values, calls getxy to get x, compares to desired value
      common/getx2pass/etapass,xpass
*
      call getxy(mrst,etapass,xiout,xout,yout,mstout)
      getx2 = xout-xpass
      return
      end

*  FUNCTION FINDMRST
* finds the mrst on a given streamline given x
      function findmrst(etastr,x,lsuper1)
      real mrst,mrstlow,mrsthigh,mrstmax,mrstbeg,mrstend
      logical succes,lsuper,lsuper1
      external getx2
      common/ends/mrstbeg,mrstend,thetaa,xend,thetanow,mrstnow,xbeg
*       !mrst at entry and exit, set in MRSTBAE, theta at entry and exit
      common/sonic/lsuper
      common/getx2pass/etapass,xpass
      common/input/rs,c1,gam  !input values for passing to subroutines
      data eps/1.0e-5/,small/0.3/
*
      mrstmax = sqrt((gam+1.0)/(gam-1.0))
      lsuper = lsuper1
      etapass = etastr  !passing to getx2
      xpass = x
      if (lsuper) then
        mrstlow = 1.0+eps
        mrsthigh = mrstend+small  !small change from last value,
* if value too big, problem with h's
        if (mrsthigh .gt. mrstmax-eps) then
          write (*,*) 'FINDMRST: mrstend is high- ',mrstend
          mrsthigh = mrstmax-eps
        end if
      else
        mrstlow = mrstbeg  !else will have problem with h's in itermrst
        mrsthigh = 1.0-eps
      end if
      xzlow = getx2(mrstlow)
      xzhigh = getx2(mrsthigh)
      if (xzlow*xzhigh .lt. 0.0) then
        succes = .true.
      else
        succes = .false.
```

```
      end if
* remove zbrac, goes off too far
*     call zbrac(getx2,mrstlow,mrsthigh,succes)
      if (succes) then
        write (*,*) 'FINDMRST: bracketed x with mrst in ',
     >        mrstlow,mrsthigh
      else
        stop 'FINDMRST: failed to bracket x'
      end if
      mrst = rtbis(getx2,mrstlow,mrsthigh,eps)
* make one more call to set up final params
      xzero = getx2(mrst)  !last call to check
      write (*,*) 'FINDMRST: x conv. within: ',xzero
      findmrst = mrst
*
      return
      end

* SUBROUTINE MSTAR:

*     The purpose of this subroutine is to calculate Eqn #17 given
*     a specific value of mrstar on the axis and a value for a
*     streamline eta.  The maximum value of eta is found by calling
*     the subroutine etawall.  If igetgrid is 1, then
*     makes an array of data for contour plots:
*     Then, at each point xi along the axis,
*     the values for hr and mr and their derivatives are calculated.
*     Using this information, eta is incremented from the axis (eta=0)
*     to the wall value (eta=etaw).  At each one of these points, the
*     corresponding value for x, y, theta, and mrstar is calculated.
*
*     These are the variables used in this subroutine:
*
*     hr - The value of h along the reference line (nozzle axis).
*          Defined in Eqn. #12.
*     mr - The value of the Mach nuber along the reference line.
*     &&pr - The primed value of the function (either hr, mr, mrst, or theta3)
*     &&dpr - The second derivative of the function (either hr or mr)
*     hrtpr - The third derivative of hr
*     hrfpr - The fourth derivative of hr
*     xi - Velocity potential with units of length
*     etaw - The value of eta which corresponds to the wall
*     eterm - Epxonential of Eqn #22
*     x2 - Second term in series for x.  Defined in Eqn #11a, #17a
*     x4 - Fourth term in series for x.  Defined in Eqn #11a, #17a
*     y1 - First term in series for y.  Defined in Eqn #11b, #17b
*     y3 - Third term in series for y.  Defined in Eqn #11b, #17b
*     theta1 - First term in series for theta.  Defined in Eqn #11c, #17c
*     theta3 - Third term in series for theta.  Defined in Eqn #11c, #17c
*     q2 - Second term in series for mstar.  Defined in Eqn #11d, #17d
*     q4 - Fourth term in series for mstar.  Defined in Eqn #11d, #17d
*     eta - The value of eta used from eta = 0 to eta = etaw.
*     mrst - The value of mstar on the reference line.
*     ra - The actual radius of curvature input by the user
*     c1 - Reference bounday constant as defined in Eqn #34
*     rs - Reference bounday radius of curvature as defined in Eqn #33
*     thetaa - The actual initial flow angle of the wall input by
*              the user.
*     dmrdmrst - Derivative of mr wrt mrstar for mrpr calculation
*     dmrstdhr - Derivative of mrstar wrt hr for mrpr calculation
```

```
*     mrstbeg - The initial value of mrst determined from thetaa in mrstbae
*     mrstend - The final value of mrst determined from xend in mrstbae
*     dmrst - Delta mrst for the iteration in the direction of constant xi
*     eden - The denominator for the exponential term
*     deleta - Delta eta for the iteration in the direction of constant eta
*     ddmrdmrst - Derivative of dmrdmrst wrt xi for mrdpr calculation
*     ddmrstdhr - Derivative of dmrstdhr wrt xi for mrdpr calculation
*     xisl0 - The value of xi where dy/dx = 0 for shifting plots so minimum
*             point on top streamline is located at xi = 0.
*     ddc1 - Constant for calculation of mr derivatives
*     ddc& - Constants 2-4 for calculation of ddmrstdhr
*     thc& - Constants 1-5 for calculation of theta3pr
*     imax - The maximum number of points in the xi direction
*     jmax - The maximum number of points in the eta direction
*     gam - The ratio of specific heats
*     gmm - gam - 1.
*     gmp - gam + 1.
*
*     Define the variables used in this subroutine and call etawall
*     in order to find the eta which corresponds to the wall value.
*
      subroutine mstar
      parameter (igetgrid=0, imax = 2, jmax = 2)
* only fill grid if igetgrid=1, then make imax,jmax nonzero
      real c1, rs, gmm, gmp, gam
      real mrstbeg, mrstend, mdc4pr, dmrst, mrstnow
      real xi, hr, hrpr, hrdpr, hrtpr, hrfpr, deleta, q4, etaw
      real mr, dmrdmrst, xend, thetaa, mrst, eta, eterm, eden
      real ddc1, dmrstdhr, mrpr, mrstpr, ddmrdmrst, mdc2, mdc3
      real mdc4, mdc1, mdc1pr, mdc2pr, mdc3pr, mrdpr, thc1, thc2, thc3
      real thc4, thc5, theta3pr, theta1, theta3, x2, x4, y1, y3, q2
      real x(imax,jmax),y(imax,jmax),theta(imax,jmax),machst(imax,jmax)
      real thc1pr, thc2pr, thc3pr, thc4pr, thc5pr, thc6pr
      real xsl0, label(8), mrstthroat, mrstmax
      integer i, j
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/throat/etaw,xithroat,mrstthroat,xsl0 !results at wall,throat from
* first iteration for continuation. Note that x,y = 0,1 at throat!
* remember that mrst depends only on xi, not on eta!
      common/ends/mrstbeg,mrstend,thetaa,xend,thetanow,mrstnow,xbeg
*
      gmm = gam-1.0
      gmp = gam+1.0
      if (igetgrid .eq. 1) then
        open(2, file='hophill.in')
        write(2,*)'TITLE="Countours in Throat Region"'
        write(2,*)'VARIABLES="X","Y","THETA","M*","XI","ETA"'
        write(2,*)'ZONE T="Contours",I=',imax,',J=',jmax,',F=POINT'
      end if
      call etawall  !returns values in the common block param (etaw,xsl0)
      write (*,*) 'Eta for the wall =',etaw
      call mrstbae !returns values in common block /ends/
      write (*,*) 'The initial Mr* =', mrstbeg
      write (*,*) 'The final Mr* =', mrstend

      if (igetgrid .eq. 1) then
      dmrst = (mrstend - mrstbeg)/FLOAT(imax-1)
      if (dmrst .le. 0.0) stop 'MSTAR: dmrst eq 0, fatal'
*
*     Set the number of points in the eta direction to be 100.  This can
```

```
*     be changed by changing the denomenator on the deleta term.
*     Then set constants and the initial integer counter in the x direction.
*     Finally, enter the do loop to move along in the x direction on
*     axis (eta = 0).
*
      deleta = etaw/FLOAT(jmax-1)
      eden = 2.*rs*c1
      z = .5
      do 5 k = 1,8
        label(k) = z
        z=z+.1
5     continue
      mrstmax = sqrt(gmp/gmm)
      do 10 i = 1,imax
        mrst = mrstbeg+(i-1)*dmrst
        if (mrst .le. 0) stop 'MSTAR: mrst le 0, fatal'
        if (mrst .ge. mrstmax) stop 'MSTAR: mrst ge mrstmax, fatal'
*
*     Calculate hr and its derivatives and mr and its derivatives.  Also
*     calculate the value of xi corresponding to the given value for
*     mrstar from the do loop.
*
        hr = SQRT((1./mrst)*(0.5*gmp - 0.5*gmm*mrst**2)**(-1./(gmm)))
        xi = SQRT(2.*rs*c1*LOG(c1/(c1 - hr + 1.)))
        if(mrst .LE. 1.) then
          xi = -xi
        endif
        eterm = EXP(-xi**2/eden)
        hrpr = (xi/rs)*eterm
        hrdpr = (1./rs)*eterm - (xi**2/(rs**2*c1))*eterm
        hrtpr = -(3.*xi/(rs**2*c1))*eterm + (xi**3./(rs**3*c1**2))*
     +           eterm
        hrfpr = -(3./(rs**2*c1))*eterm + (6.*xi**2/(rs**3*c1**2))*
     +          eterm - (xi**4/(rs**4*c1**3))*eterm
        mr = SQRT(2*mrst**2/(gmp - gmm*mrst**2))
        ddc1 = 0.5*gmp - 0.5*gmm*mrst**2
        dmrdmrst = (2.*mrst*gmp/(mr*(gmp - gmm*mrst**2)**2))
        if(mrst .LT. 0.99 .OR. mrst .GT. 1.01) then
           dmrstdhr = 2./(mrst*hr/ddc1 - hr/mrst)
           mrpr = dmrdmrst*dmrstdhr*hrpr
           mrstpr = mrpr/dmrdmrst
           ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +              2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
           mdc1 = gmp*mr
           mdc1pr = gmp*mrpr
           mdc2= mrst**2
           mdc2pr = 2.*mrst*mrstpr
           mdc3 = hr
           mdc3pr = hrpr
           mdc4 = hr/mr**2
           mdc4pr = (hrpr*mr - 2.*hr*mrpr)/(mr**3)
           mrdpr = ((mdc1pr*(mdc2*(mdc3-mdc4))-mdc1*(mdc2pr*(mdc3-mdc4)+
     +             mdc2*(mdc3pr-mdc4pr)))/((mdc2*(mdc3-mdc4))**2))*hrpr+
     +             dmrdmrst*dmrstdhr*hrdpr
        else
           mrpr = dmrdmrst*SQRT(2./(gmp*rs))
           mrstpr = mrpr/dmrdmrst
           ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +              2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
           mrdpr = ddmrdmrst*SQRT(2./(gmp*rs))
```

```
          endif
*
*     Calculate the values for theta1, theta3, theta3pr, y1, y3, x2,
*     x4, q2, q4 from Eqn #17a-#17d.  All of these remain constant at
*     any eta for a given value of xi since they are only dependant
*     on hr, mr, and their derivatives.
*
          thc1 = hrdpr
          thc1pr = hrtpr
          thc2 = hr*hrpr*mr**2
          thc2pr = (hrpr*mr)**2 + hr*hrdpr*mr**2 + 2*hr*hrpr*mr*mrpr
          thc3 = hr**2*mr*mrpr
          thc3pr = 2*hr*hrpr*mr*mrpr + (hr*mrpr)**2 + hr**2*mr*mrdpr
          thc4 = hr**2*hrtpr
          thc4pr = 2*hr*hrpr*hrtpr + hr**2*hrfpr
          thc5 = mr**2-1
          thc5pr = 2*mr*mrpr
          thc6pr = hrpr**2*hrdpr
          theta3pr = 0.25*(thc1pr*(thc2+thc3) + thc1*(thc2pr+thc3pr)) +
     +               0.125*(thc4pr*thc5 + thc5pr*thc4 + thc6pr)
          theta1 = hrpr
          theta3 = (1./4.)*(hrdpr*(hr*hrpr*mr**2 + hr**2*mr*mrpr)) +
     +             (1./8.)*hr**2*hrtpr*(mr**2-1.) + (1./24.)*hrpr**3
          x2 = 0.5*hr*hrpr
          x4 = (3./32)*(hr**2*hrpr*hrdpr*(mr**2-1)) - (1./96.)*(hr*
     +         hrpr**3) + theta3*hr*0.25
          y1 = hr
          y3 = (hr/8)*(hr*hrdpr*(mr**2-1) - hrpr**2)
          q2 = 0.5*hr*hrdpr
          q4 = 0.25*(hr**2*hrdpr**2) + (3./32.)*(hr**2*hrdpr**2*(mr**2-
     +         1)) + (1/32.)*(hr*hrdpr*hrpr**2) + 0.25*hr*theta3pr
          x2pr = (0.5*(hrpr**2 + hr*hrdpr))
          x4c1 = (3./32.)*((mr**2-1)*(2.*hr*hrpr**2*hrdpr +hr**2*
     +           hrdpr**2+hr**2*hrpr*hrtpr)+2.*hr**2*hrpr*hrdpr*mr*mrpr)
          x4c2 = (1./96.)*(hrpr**4 + 3.*hr*hrpr**2*hrdpr)
          x4c3 = 0.25*(theta3pr*hr + theta3*hrpr)
          x4pr = (x4c1 - x4c2 + x4c3)
          y1pr = hrpr
          y3pr = (hrpr/8.)*(hr*hrdpr*(mr**2-1) - hrpr**2) + (hr/8.)*
     +           (hrpr*hrdpr*(mr**2-1) + hr*hrtpr*(mr**2-1) + 2*hr*
     +           hrdpr*mr*mrpr - 2*hrpr*hrdpr)

*
*     Set the counter value for the arrays for different eta (0->etaw).
*     Then enter the do loop for the calculation of Eqn #17.
*
          eta = 0.
          do 20 j = 1, jmax
*
*     Calculate Eqn #17a.  Then, if mrstar is less than one, make the value
*     for x the negative value for plotting purposes.  This is because
*     the value for xi is always positive along the axis.  However, since
*     the paper makes the assumption that the sonic line on the axis occurs
*     at x = 0 (pp. 1339 paragraph before section #3, and figure #3), it
*     is known that for all values of mrstar less than 1, the x value
*     should be negative.
*
             x(i,j) = xi - x2*eta**2 - x4*eta**4
*
*     Shift the x axis by xsl0 so that the point where x = 0 corresponds
```

```
*     to the nozzle throat.
*
           x(i,j) = x(i,j) - xsl0
*
*     Calculate Eqn #17b.
*
           y(i,j) = y1*eta + y3*eta**3
*
*     Calculate Eqn #17c.
*
           theta(i,j) = ATAN((y1pr*eta+y3pr*eta**3)/(1.-x2pr*eta**2 -
     +                    x4pr*eta**4))
*
*     Calculate Eqn #17d.
*
           machst(i,j) = mrst*(1 + q2*eta**2 + q4*eta**4)
*
*     Change counters for the next values.
*
           write(2,99) x(i,j),y(i,j),theta(i,j),machst(i,j),xi,eta
           eta = eta + deleta
20      continue
10    continue
      close(2)
      end if
99    Format(6(1x,e14.7))
      return
      end


* SUBROUTINE ETAWALL (name changed from etamax sps 6-2-93)

*     This subroutine is used to find the streamline which will
*     correspond to the contour for the wall of the nozzle.  This is
*     done using a bisection method to iterate of xi to find the
*     value of eta where dydx = 0 and y = 1 as defined by the paper
*     in section 4, pp 1340.
*
*     The variables that are used in this subroutine are listed below:
*
*     hr - The value of h along the reference line (nozzle axis).
*          Defined in Eqn. #12.
*     mr - The value of the Mach nuber along the reference line.
*     &&pr - The primed value of the function (either hr, mr, y1, or y3)
*     hrdpr - The second derivative of hr
*     hrtpr - The third derivative of hr
*     xi - Velocity potential with units of length
*     etaw - The value of eta which corresponds to the wall
*     eterm - Epxonential of Eqn #22
*     eden - The denominator for the exponential term
*     x2 - Second term in series for x.  Defined in Eqn #11a, #17a
*     x4 - Fourth term in series for x.  Defined in Eqn #11a, #17a
*     y1 - First term in series for y.  Defined in Eqn #11b, #17b
*     y3 - Third term in series for y.  Defined in Eqn #11b, #17b
*     theta3 - Third term in series for theta.  Defined in Eqn #11c, #17c
*     mrst - The value of mstar on the reference line.
*     c1 - Reference bounday constant as defined in Eqn #34
*     rs - Reference bounday radius of curvature as defined in Eqn #33
*     dmrdmrst - Derivative of mr wrt mrstar for mrpr calculation
*     dmrstdhr - Derivative of mrstar wrt hr for mrpr calculation
```

```
*     y - y location of point where dydx = 0.  Defined in Eqn #11b, #17b
*     x - x location of point where dydx = 0.  Defined in Eqn #11a, #17a
*     xihigh - The upper bounds for bisection method in xi
*     xilow - The lower bounds for bisection method in xi
*     ddc1 - The constant used in the calculation of mrpr
*     gam - The ratio of specific heats
*     gmm - gam - 1.
*     gmp - gam + 1.
*     eps - The convergence criteria for the iteration.
*
*     The first part of the subroutine sets all of the variables used
*     in this subroutine and prints that the program is seraching for
*     the eta that corresponds to the wall contour.
*

      subroutine etawall
      real gam, c1, rs, xihigh, xilow
      real eps, xsl0, mrstthroat, mstth
      external yminm,rtbis !declares it an external function, needed for NR call
      logical succes,lsuper
      common/sonic/lsuper  !tells subroutine if mrst (Not mst!) in sonic area
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/throat/etaw,xithroat,mrstthroat,xsl0
*                  !save for later comparisons, value of eta
*                        !at the nozzle wall, y=1. sps
* remember that mrst depends only on xi, not on eta.  So mrst = mrst
* at same xi
      data tiny/2.0e-7/
      data eps/1.0e-5/ !reduced from 0.00035 by sps with new routine 6-2-93
*
      write (*,*) 'Search for Eta on Contour'
*
*     This part of the program sets the denominator for the exponential
*     and sets the limits on the maximum and minimum values for xi
*     for the iteration.  Since the value for dydx = 0 is near the axis,
*     the high value needs to be small.  If it is not, the program will
*     not give accurate results.  As the radius of curvature gets smaller,
*     the high value must get closer to zero.  Since the only xi that is
*     used is the squared value, the low value should always be left at
*     zero to avoid division by zero in the dmrstdhr term when xi = 0.
*
      xihigh = -0.03
      xilow = -0.2
*
* Call Numerical Recipes Routine to bracket for the root
      call zbrac(yminm,xilow,xihigh,succes)
      if (succes) then
        write (*,*) 'ETAWALL: successfully bracketed root in ',
     >                                                 xilow,xihigh
      else
        stop 'ETAWALL: failed to bracket root'
      end if
*
* Call Numerical Recipes Routine to iterate via Bisection for root
*
      xith1 = rtbis(yminm,xilow,xihigh,eps)
*  xith1 should be xi at the throat to within eps, now
*  function yminm also sets the value of parameters in common block /throat/
*  BUT, last call from rtbis may not be at converged value.  So make
* this last call:
```

```
      ythroat = yminm(xith1)
      write (*,*) 'ETAWALL: ythroat converged to: ',ythroat
*
      if (abs(xith1-xithroat) .gt. tiny) then
        write (*,*) 'ETAWALL: xith1,xithroat= ',xith1,xithroat
        stop 'ETAWALL: passing error'
      end if
      call getxy(mrstthroat,etaw,xith1,xth,yth,mstth)
      write (*,*) '************************************'
      write (*,*) 'ETAWALL: set common block /throat/:'
      write (*,*) 'etaw=', etaw,' xithroat= ',xithroat
      write (*,*) 'mrstthroat= ',mrstthroat,' xsl0= ',xsl0
      write (*,*) 'GETXY gives throat value for mstth= ',mstth
      write (*,*) 'and x,y at physical throat, wall= ',xth,yth
      write (*,*) '************************************'
      return
      end

*  FUNCTION GETX
* this is a function for ETAWALL that contains iterated function
*   part of the solver for the throat values of eta and xi.
*   modified from YMINM by sps
*   added sps 6-2-93 to allow use of Numerical Recipes bisector code
*
      function getx(xi)
      real eterm, eden, hr, hrpr, hrdpr, etaw, y1pr, y3pr, y1
      real hrtpr, gmm, gmp, gam, c1, rs, y3, xi
      real xsl0, mrst, mr, dmrdmrst, ddc1, dmrstdhr, mrpr
      real mrstthroat,mrstpass
      logical lsuper !true if iterating for mrst (Not mst!) in supersonic region
      common/sonic/lsuper
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/throat/etaw,xithroat,mrstthroat,xsl0 !values at throat
      common/getxpass/xpass,etapass,mrstpass
* (only used to pass to this subroutine)
*
      gmm = gam-1.0
      gmp = gam+1.0
*
      eden = 2.*rs*c1
*
* set sonic branch, see top rh column p. 1339
      if (xi .lt. 0) then
*          !mrst subsonic, xi=0 is sonic line, exactly, for mrst  (not mst!)
        lsuper = .false.
      else
        lsuper = .true.
      end if
* Note that fig. 4, others, show supersonic flow at throat.
* BUT THIS IS MST, NOT!! MRST WHICH IS WHAT IS BEING DECIDED HERE!
*     write (*,*) 'GETX: lsuper = ',lsuper,',if this false, problem?'
*
      eterm = EXP(-xi**2/eden)
      hr = c1*(1-eterm) + 1.
      hrpr = (xi/rs)*eterm
      hrdpr = (1./rs)*eterm - (xi**2/(rs**2*c1))*eterm
      hrtpr = -(3.*xi/(rs**2*c1))*eterm + (xi**3./(rs**3*c1**2))*
     +        eterm
*
*     Call subroutine to determine the value of mrstar given hr.
```

```
*
      call itermrst(hr, mrst)
* Recursive call to rtbis inside itermrst! rename this call rtbis2!!
*
*     Calculate the value of mr and the needed derivatives.
*
      mr = SQRT(2.*mrst**2/(gmp - gmm*mrst**2))
      dmrdmrst = (2.*mrst*gmp/(mr*(gmp - gmm*mrst**2)**2))
      ddc1 = 0.5*gmp - 0.5*gmm*mrst**2
      if(mrst .LT. 0.99 .OR. mrst .GT. 1.01) then
        dmrstdhr = 2./(mrst*hr/ddc1 - hr/mrst)
        mrpr = dmrdmrst*dmrstdhr*hrpr
      else
        mrpr = dmrdmrst*SQRT(2./(gmp*rs))
      endif
*
*     Calculate the values for the constants in the y equation (#11b,
*     #17b) and their derivatives.
*
      y1 = hr
      y3 = (hr/8.0)*(hr*hrdpr*(mr**2-1) - hrpr**2)
      y1pr = hrpr
      y3pr = (hrpr/8.)*(hr*hrdpr*(mr**2-1) - hrpr**2) + (hr/8.)*
     +       (hrpr*hrdpr*(mr**2-1) + hr*hrtpr*(mr**2-1) + 2*hr*
     +       hrdpr*mr*mrpr - 2*hrpr*hrdpr)
*
*
* Now compute some other things also will be needed if this is root
      theta3 = (1./4.)*(hrdpr*(hr*hrpr*mr**2 + hr**2*mr*mrpr)) +
     >           (1./8.)*hr**2*hrtpr*(mr**2-1.) + (1./24.)*hrpr**3
      x2 = 0.5*hr*hrpr
      x4 = (3./32)*(hr**2*hrpr*hrdpr*(mr**2-1)) - (1./96.)*(hr*
     >   hrpr**3) + theta3*hr*0.25
      x = xi - x2*etapass**2 - x4*etapass**4
      x = x - xsl0  !shift so x=0 at throat
      getx = x-xpass  !return a value that should be zero, for rtbis
      mrstpass = mrst
*     write (*,5) xi,x,getx,mrstpass
*   5 format(' GETX: return xi,x,getx,mrstpass= ',4(f10.6,1x))
*     write (*,*) 'and xpass was ',xpass
*
      return
      end


*  FUNCTION YMINM
* this is a function for ETAWALL that contains iterated function
*   part of the solver for the throat values of eta and xi.
*   modified from ETAWALL by sps
*   added sps 6-2-93 to allow use of Numerical Recipes bisector code
*
      function yminm(xi)
      real eterm, eden, hr, hrpr, hrdpr, etaw, y1pr, y3pr, y1
      real hrtpr, gmm, gmp, gam, c1, rs, y3, xi
      real xsl0, mrst, mr, dmrdmrst, ddc1, dmrstdhr, mrpr
      real mrstthroat
      logical lsuper  !true if iterating for mrst in supersonic region
      common/sonic/lsuper
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/throat/etaw,xithroat,mrstthroat,xsl0 !values at throat
```

```
*
      gmm = gam-1.0
      gmp = gam+1.0
*
      eden = 2.*rs*c1
*
* set sonic branch, see top rh column p. 1339, Note this is mrst, not mst!!
      if (xi .lt. 0.0) then  !subsonic, xi=0 is sonic line for msrt
        lsuper = .false.
      else
        lsuper = .true.
      end if
* Note that fig. 4, others, show supersonic flow at throat.  But this
* is mst.  Here we are looking at mrst, which is set by xi alone
*     write (*,*) 'YMINM: lsuper = ',lsuper,',if this false, problem?'
*
      eterm = EXP(-xi**2/eden)
      hr = c1*(1-eterm) + 1.
      hrpr = (xi/rs)*eterm
      hrdpr = (1./rs)*eterm - (xi**2/(rs**2*c1))*eterm
      hrtpr = -(3.*xi/(rs**2*c1))*eterm + (xi**3./(rs**3*c1**2))*
     +        eterm
*
*     Call subroutine to determine the value of mrstar given hr.
*
      call itermrst(hr, mrst)
* Recursive call to rtbis inside itermrst! rename this call rtbis2!!
*
*     Calculate the value of mr and the needed derivatives.
*
      mr = SQRT(2.*mrst**2/(gmp - gmm*mrst**2))
      dmrdmrst = (2.*mrst*gmp/(mr*(gmp - gmm*mrst**2)**2))
      ddc1 = 0.5*gmp - 0.5*gmm*mrst**2
      if(mrst .LT. 0.99 .OR. mrst .GT. 1.01) then
        dmrstdhr = 2./(mrst*hr/ddc1 - hr/mrst)
        mrpr = dmrdmrst*dmrstdhr*hrpr
      else
        mrpr = dmrdmrst*SQRT(2./(gmp*rs))
      endif
*
*     Calculate the values for the constants in the y equation (#11b,
*     #17b) and their derivatives.
*
      y1 = hr
      y3 = (hr/8.0)*(hr*hrdpr*(mr**2-1) - hrpr**2)
      y1pr = hrpr
      y3pr = (hrpr/8.)*(hr*hrdpr*(mr**2-1) - hrpr**2) + (hr/8.)*
     +       (hrpr*hrdpr*(mr**2-1) + hr*hrtpr*(mr**2-1) + 2*hr*
     +       hrdpr*mr*mrpr - 2*hrpr*hrdpr)
*
*     Set the value for eta where dydx = 0 as given by equation #25 and
*     calculate the value for y at that location.
*
      argeta = -y1pr/y3pr
      if (argeta .gt. 0.0) then
        etaw = SQRT(-y1pr/y3pr)
      else
        write (*,*) 'YMINM: argeta = ',argeta,' lt 0'
        write (*,*) 'for xi = ',xi
        stop 'fatal error, need sqrt of this'
```

```
      end if
      yminm = y1*etaw + y3*etaw**3 - 1.0
* program converges if ymin = 1.0, so subtract 1 here to provide zero
*   seeking function to Numerical Recipes
*  write out for debug
*     write (*,5) xi,etaw,yminm
*   5 format(' YMINM: return xi,etaw,ymin-1= ',3(f10.6,1x))
*
* Now compute some other things also will be needed if this is root
      theta3 = (1./4.)*(hrdpr*(hr*hrpr*mr**2 + hr**2*mr*mrpr)) +
     >         (1./8.)*hr**2*hrtpr*(mr**2-1.) + (1./24.)*hrpr**3
      x2 = 0.5*hr*hrpr
      x4 = (3./32)*(hr**2*hrpr*hrdpr*(mr**2-1)) - (1./96.)*(hr*
     >   hrpr**3) + theta3*hr*0.25
      xsl0 = xi - x2*etaw**2 - x4*etaw**4
      xithroat = xi    !these two added sps for continuation later
      mrstthroat = mrst
*
      return
      end



*  SUBROUTINE MRSTBAE
* finds the mrst at the beginning angle and ending x
      subroutine mrstbae
      real mrstbeg,mrstend,mrstnow,mstout,mrstpass
      logical succes,lsuper
      external anglezer,rtbis,golden,mnbrak,getx
      common/sonic/lsuper
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/ends/mrstbeg,mrstend,thetaa,xend,thetanow,mrstnow,xbeg
      common/throat/etaw,xithroat,mrstthroat,xsl0 !results at wall,throat from
* first iteration for continuation. Note that x,y = 0,1 at throat!
* remember that mrst depends only on xi, not on eta!
      common/getxpass/xpass,etapass,mrstpass
* (only used to pass to getx)
      data eps/1.0e-5/
*     data nplt/50/
      data small/0.01/!entry angle, closeness to specified value
*
      thetanow = thetaa  !do entry first
      xilow = -10.0
      xihigh = 0.0
      lsuper = .false.  !subsonic area, needed for mrst branch
      anglow = anglezer(xilow)
      anghigh = anglezer(xihigh)
      if (anglow*anghigh .lt. 0.0) then
        succes = .true.
      else
        succes = .false.
      end if
* remove zbrac, takes off too far looking for bracket
*     call zbrac(anglezer,xilow,xihigh,succes)
      if (succes) then
        write (*,*) 'MRSTBAE:bracketed entry angle, xi in ',xilow,xihigh
        write (*,*) 'corresponding angles are: ',
     >     anglezer(xilow)+thetanow,anglezer(xihigh)+thetanow
        xibeg = rtbis(anglezer,xilow,xihigh,eps)
* make one more call to set up final params
```

```
        aangzero = anglezer(xibeg)  !last call to set up other params
        write (*,*) 'MRSTBAE: entry angle conv. within: ',aangzero
        mrstbeg = mrstnow
        write (*,*) 'MRSTBAE: xibeg,mrstbeg= ',xibeg,mrstbeg
      else !go for the minimum
        write (*,*) 'xilow,xihigh guessed as ',xilow,xihigh
        write (*,*) 'corresp. angles are: ',
     >   anglezer(xilow)+thetanow,anglezer(xihigh)+thetanow
        write (*,*) 'entry angle is ',thetaa
        write (*,*) 'MRSTBAE: failed to bracket, find minimum'
*       if (xilow .lt. -100) xilow=-20  !don't go way out
*       delxi = (xihigh-xilow)/float(nplt-1)
*       write (*,*) 'writing plot data to "entryang.deb"'
*       open (unit=9,file='entryang.deb')
*       write (9,*) 'table of xi,angle(xi): '
*       do 50 i = 1,nplt
*         xi = xilow+(i-1)*delxi
*         write (9,*) xi,anglezer(xi)+thetanow
*  50   continue
*       close (unit=9)
* Call numerical recipes routine to find the minimum
        ax = -8.0
        bx = -5.0 !a guess near minimum
        cx = -3.0
        if (anglezer(bx) .lt. anglezer(ax) .and.
     >       anglezer(bx) .lt. anglezer(cx)) then
          write (*,*) 'minimum bracketed'
        else
          call mnbrak(ax,bx,cx,fa,fb,fc,anglezer)
        end if
        anglemin = golden(ax,bx,cx,anglezer,eps,ximin)+thetanow
        write (*,*) 'minimum entry angle is ',anglemin
        write (*,*) 'located at ximin= ',ximin
        if ((anglemin-thetaa)/thetaa .lt. small) then
          write (*,*) 'close to entry angle, keep this one'
          mrstbeg = mrstnow
          xibeg = ximin
          write (*,*) 'MRSTBAE: xibeg,mrstbeg= ',xibeg,mrstbeg
        else
          stop 'MRSTBAE: failed to find entry angle'
        end if
      end if
      call getxy(mrstbeg,etaw,xiout,xout,yout,mstout)
      xbeg = xout
      write (*,*) 'MRSTBAE: xbeg= ',xbeg
*
      write (*,*) 'MRSTBAE: now finding exit xi on contour: '
*
      etapass = etaw  !passing to getx
      xpass = xend
      xilow = -1.0
      xihigh = 8.0
      xzlow = getx(xilow)
      xzhigh = getx(xihigh)
      if (xzlow*xzhigh .lt. 0.0) then
        succes = .true.
      else
        succes = .false.
      end if
* remove zbrac, goes off too far
```

```
*     call zbrac(getx,xilow,xihigh,succes)
      if (succes) then
        write (*,*) 'MRSTBAE: bracketed xend with xi in ',xilow,xihigh
      else
        write (*,*) 'MRSTBAE: xend,xzhigh,xzlow= ',xend,xzhigh,xzlow
        write (*,*) ' and xilow,xihigh= ',xilow,xihigh
        stop 'MRSTBAE: failed to bracket xend'
      end if
      xiend = rtbis(getx,xilow,xihigh,eps)
* make one more call to set up final params
      xzero = getx(xiend)  !last call to set up other params
      write (*,*) 'MRSTBAE: xend conv. within: ',xzero
      mrstend = mrstpass  !passed back from getx
      write (*,*) 'MRSTBAE: xiend,mrstend= ',xiend,mrstend
*
      return
      end

*  FUNCTION ANGLEZER
* adapted by sps from MRSTBAE to contain function for Numerical Recipes
*   iteration, 6-2-93
* finds the mrst at the beginning and ending angles

      function anglezer(xi)
      real thetaa, c1, rs, gam, gmp, gmm, etaw
      real mrstbeg, mrstend, mrstnow
      real xi, eterm, eden, hr, hrpr, hrdpr, thetacalc
      real hrtpr, mrst, mr, dmrdmrst, ddc1, dmrstdhr, mrpr
      real hrfpr, mdc1, mdc2, mdc3, mdc4, mdc1pr, theta3
      real mrdpr, mdc2pr, mdc3pr, mdc4pr
      logical lsuper  !true if iterating for mrst in supersonic region
      common/sonic/lsuper
      common/input/rs,c1,gam  !input values for passing to subroutines
      common/ends/mrstbeg,mrstend,thetaa,xend,thetanow,mrstnow,xbeg
      common/throat/etaw,xithroat,mrstthroat,xsl0 !results at wall,throat from
* first iteration for continuation. Note that x,y = 0,1 at throat!
* remember that mrst depends only on xi, not on eta!
*
      gmm = gam-1.0
      gmp = gam+1.0
      eden = 2.*rs*c1
      eterm = EXP(-xi**2/eden)
      hr = c1*(1-eterm) + 1.
      hrpr = (xi/rs)*eterm
      hrdpr = (1./rs)*eterm - (xi**2/(rs**2*c1))*eterm
      hrtpr = -(3.*xi/(rs**2*c1))*eterm + (xi**3./(rs**3*c1**2))*
     +        eterm
      hrfpr = -(3./(rs**2*c1))*eterm + (6.*xi**2/(rs**3*c1**2))*
     +       eterm - (xi**4/(rs**4*c1**3))*eterm
      call itermrst(hr, mrst)  !finds mrst given hr
      mr = SQRT(2*mrst**2/(gmp - gmm*mrst**2))
      ddc1 = 0.5*gmp - 0.5*gmm*mrst**2
      if(mrst .LT. 0.99 .OR. mrst .GT. 1.01) then
        dmrdmrst = (2.*mrst*gmp/(mr*(gmp - gmm*mrst**2)**2))
        dmrstdhr = 2./(mrst*hr/ddc1 - hr/mrst)
        mrpr = dmrdmrst*dmrstdhr*hrpr
        mrstpr = mrpr/dmrdmrst
        ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +           2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
        mdc1 = gmp*mr
```

```
          mdc1pr = gmp*mrpr
          mdc2= mrst**2
          mdc2pr = 2.*mrst*mrstpr
          mdc3 = hr
          mdc3pr = hrpr
          mdc4 = hr/mr**2
          mdc4pr = (hrpr*mr - 2.*hr*mrpr)/(mr**3)
          mrdpr = ((mdc1pr*(mdc2*(mdc3-mdc4))-mdc1*(mdc2pr*(mdc3-mdc4)+
     +           mdc2*(mdc3pr-mdc4pr)))/((mdc2*(mdc3-mdc4))**2))*hrpr+
     +           dmrdmrst*dmrstdhr*hrdpr
        else
          mrpr = dmrdmrst*SQRT(2./(gmp*rs))
          mrstpr = mrpr/dmrdmrst
          ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +         2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
          mrdpr = ddmrdmrst*SQRT(2./(gmp*rs))
        endif
        theta3 = (1./4.)*(hrdpr*(hr*hrpr*mr**2 + hr**2*mr*mrpr)) +
     +          (1./8.)*hr**2*hrtpr*(mr**2-1.) + (1./24.)*hrpr**3
        thc1 = hrdpr
        thc1pr = hrtpr
        thc2 = hr*hrpr*mr**2
        thc2pr = (hrpr*mr)**2 + hr*hrdpr*mr**2 + 2*hr*hrpr*mr*mrpr
        thc3 = hr**2*mr*mrpr
        thc3pr = 2*hr*hrpr*mr*mrpr + (hr*mrpr)**2 + hr**2*mr*mrdpr
        thc4 = hr**2*hrtpr
        thc4pr = 2*hr*hrpr*hrtpr + hr**2*hrfpr
        thc5 = mr**2-1
        thc5pr = 2*mr*mrpr
        thc6pr = hrpr**2*hrdpr
        theta3pr = 0.25*(thc1pr*(thc2+thc3) + thc1*(thc2pr+thc3pr)) +
     +           0.125*(thc4pr*thc5 + thc5pr*thc4 + thc6pr)
        x2pr = (0.5*(hrpr**2 + hr*hrdpr))
        x4c1 = (3./32.)*((mr**2-1)*(2.*hr*hrpr**2*hrdpr +hr**2*
     +        hrdpr**2+hr**2*hrpr*hrtpr)+2.*hr**2*hrpr*hrdpr*mr*mrpr)
        x4c2 = (1./96.)*(hrpr**4 + 3.*hr*hrpr**2*hrdpr)
        x4c3 = 0.25*(theta3pr*hr + theta3*hrpr)
        x4pr = (x4c1 - x4c2 + x4c3)
        y1pr = hrpr
        y3pr = (hrpr/8.)*(hr*hrdpr*(mr**2-1) - hrpr**2) + (hr/8.)*
     +        (hrpr*hrdpr*(mr**2-1) + hr*hrtpr*(mr**2-1) + 2*hr*
     +        hrdpr*mr*mrpr - 2*hrpr*hrdpr)
        thetacalc = ATAN((y1pr*etaw+y3pr*etaw**3)/(1.-x2pr*etaw**2 -
     +            x4pr*etaw**4))
        anglezer = thetacalc - thetanow  !use thetanow for current sought value
        mrstnow = mrst
*       write (*,*) 'ANGLEZER: return mrstnow,anglezer= ',
*    >    mrstnow,anglezer
*       write (*,*) 'and thetacalc,thetanow= ',thetacalc,thetanow
        return
        end


* SUBROUTINE ITERMRST


*       The purpose of this subroutine is to find a value for mrstar given
*       a value for hr.  This is done by using a bisection method to
*       iterate on mrstar to find the value of hr that is sent to the
```

```
*     subroutine.
*
*     The variables used in this subroutine are as follows:
*
*     hr - The value of hr sent to this subroutine that is being matched.
*     hrcalc - The calculated value of hr for a given mrstar.
*     mrst - The value for mrstar that solves (13), sought here.
*     gam - The ratio of specific heats.
*     gmp - Gamma + 1.
*     gmm - Gamma - 1.
*     mrsthigh - The upper guess on mrst
*     mrstlow - The lower guess on mrst
*
*     The first part of the subroutine defines all of the variables and
*     sets the guesses for mrst.
*
      subroutine itermrst(hr, mrst)
      real hrzero, hr, mrst, mrsthigh, mrstlow, mrstmax
      logical succes,lsuper  !true if supersonic (2 branches!!)
      external hrzero,rtbis2
      common/sonic/lsuper
      common /hriter/ hrwant  ! a common block to pass the desired value
      common/input/rs,c1,gam  !input values for passing to subroutines
      data eps/1.0e-6/
*
* if hr = 1.0 exactly, handle as a special case to avoid singularities
      if (hr .eq. 1.0) then
        mrst = 1.0
        write (*,*) 'ITERMRST: returning exact sonic case'
        return
      end if
*
      gmm = gam-1.0
      gmp = gam+1.0
      mrstmax = sqrt(gmp/gmm)
*
      hrwant = hr
      if (lsuper) then      !this is lsuper for mrst, Not local sonic (mst)
        mrstlow = 1.0 + eps
        mrsthigh = mrstmax - eps
      else
        mrstlow = eps
        mrsthigh = 1.0 - eps
      end if
* Call copies of numerical recipes routines to avoid recursive use,
*   not supported in ordinary fortran
*     call zbrac2(hrzero,mrstlow,mrsthigh,succes)  !avoid, searches too far
      hrzlow = hrzero(mrstlow)
      hrzhigh = hrzero(mrsthigh)
      if (hrzlow*hrzhigh .lt. 0.0) then
        succes = .true.
      else
        succes = .false.
      end if
      if (succes) then
*       write (*,*) 'ITEMMRST: hr,lsuper = ',hr,lsuper
*       write (*,*) 'ITERMRST: bracketed mrst in ',mrstlow,mrsthigh
      else
        write (*,*) 'mrstlow,mrsthigh= ',mrstlow,mrsthigh
        write (*,*) 'hrzlow,hrzhigh= ',hrzlow,hrzhigh
```

```
        write (*,*) 'hrwant= ',hrwant
        stop 'ITERMRST: failed to bracket root'
      end if
      mrst = rtbis2(hrzero,mrstlow,mrsthigh,eps)
*     write (*,*) 'ITERMRST: returned mrst= ',mrst,' at hr= ',hr,
*    >  ' lsuper= ',lsuper
*
      return
      end

* following function added by sps to allow using Numerical Recipes routines
      function hrzero(mrst)
      real mrst
      common/hriter/hrwant    !the desired value of hr
      common/input/rs,c1,gam  !input values for passing to subroutines
*
      gmm = gam-1.0
      gmp = gam+1.0
      arg1 = 0.5*gmp - 0.5*gmm*mrst**2
      if (arg1 .lt. 0.0) then
        write (*,*) 'HRZERO: arg of exp is lt 0'
        write (*,*) 'mrst,arg1= ',mrst,arg1
        stop 'fatal error'
      end if
      arg = 1./(mrst*(arg1)**(1/gmm))
      if (arg .lt. 0.0) stop 'HRZERO: arg of sqrt lt zero'
      hrcalc = SQRT(arg)
      hrzero = hrcalc - hrwant
      return
      end

* SUBROUTINE GETXY:

* This subroutine modified from MSTAR 6-2-93 by sps.  Given nozzle
* definition, and guesses for mrst and eta,
* it finds the x and y that correspond to the guesses.
* Will be called
* to find the mrst and eta for a particular
* streamline needed for the square nozzle code
*  1) this subroutine should only be called after the main program
*  has already been called, to initialize variables
*
*     The purpose of this subroutine is to calculate Eqn #17 given
*     a specific value of mrstar on the axis and a value for a
*     streamline eta.
*     Then, at each point xi along the axis,
*     the values for hr and mr and their derivatives are calculated.
*     Using this information, eta is incremented from the axis (eta=0)
*     to the wall value (eta=etaw).  At each one of these points, the
*     corresponding value for x, y, theta, and mrstar is calculated.
*
*     These are the variables used in this subroutine:
*
*     hr - The value of h along the reference line (nozzle axis).
*          Defined in Eqn. #12.
*     mr - The value of the Mach nuber along the reference line.
*     &&pr - The primed value of the function (either hr, mr, mrst, or theta3)
*     &&dpr - The second derivative of the function (either hr or mr)
*     hrtpr - The third derivative of hr
*     hrfpr - The fourth derivative of hr
```

```
*     xi - Velocity potential with units of length
*     etaw - The value of eta which corresponds to the wall
*     eterm - Epxonential of Eqn #22
*     x2 - Second term in series for x.  Defined in Eqn #11a, #17a
*     x4 - Fourth term in series for x.  Defined in Eqn #11a, #17a
*     y1 - First term in series for y.  Defined in Eqn #11b, #17b
*     y3 - Third term in series for y.  Defined in Eqn #11b, #17b
*     theta1 - First term in series for theta.  Defined in Eqn #11c, #17c
*     theta3 - Third term in series for theta.  Defined in Eqn #11c, #17c
*     q2 - Second term in series for mstar.  Defined in Eqn #11d, #17d
*     q4 - Fourth term in series for mstar.  Defined in Eqn #11d, #17d
*     eta - The value of eta used from eta = 0 to eta = etaw.
*     mrst - The value of mstar on the reference line.
*     ra - The actual radius of curvature input by the user
*     c1 - Reference bounday constant as defined in Eqn #34
*     rs - Reference bounday radius of curvature as defined in Eqn #33
*     thetaa - The actual initial flow angle of the wall input by
*              the user.
*     dmrdmrst - Derivative of mr wrt mrstar for mrpr calculation
*     dmrstdhr - Derivative of mrstar wrt hr for mrpr calculation
*     mrstbeg - The initial value of mrst determined from thetaa in mrstbae
*     dmrst - Delta mrst for the iteration in the direction of constant xi
*     eden - The denominator for the exponential term
*     deleta - Delta eta for the iteration in the direction of constant eta
*     ddmrdmrst - Derivative of dmrdmrst wrt xi for mrdpr calculation
*     ddmrstdhr - Derivative of dmrstdhr wrt xi for mrdpr calculation
*     xisl0 - The value of xi where dy/dx = 0 for shifting plots so minimum
*             point on top streamline is located at xi = 0.
*     ddc1 - Constant for calculation of mr derivatives
*     ddc& - Constants 2-4 for calculation of ddmrstdhr
*     thc& - Constants 1-5 for calculation of theta3pr
*     imax - The maximum number of points in the xi direction
*     jmax - The maximum number of points in the eta direction
*     gam - The ratio of specific heats
*     gmm - gam - 1.
*     gmp - gam + 1.
*
*     Define the variables used in this subroutine, use common block
*     in order to find the eta which corresponds to the wall value.
*
      subroutine getxy(mrst,eta,xi,x,y,mst)
*  given mrst and eta, returns xi, x, y, and mst
      real c1, rs, gmm, gmp, gam, mdc4pr
      real xi, hr, hrpr, hrdpr, hrtpr, hrfpr, q4, etaw
      real mr, dmrdmrst, mrst, eta, eterm, eden, mrstmax
      real ddc1, dmrstdhr, mrpr, mrstpr, ddmrdmrst, mdc2, mdc3
      real mdc4, mdc1, mdc1pr, mdc2pr, mdc3pr, mrdpr, thc1, thc2, thc3
      real thc4, thc5, theta3pr, theta1, theta3, x2, x4, y1, y3, q2
      real x,y,theta,mst, mrstthroat
      real thc1pr, thc2pr, thc3pr, thc4pr, thc5pr, thc6pr
      common/throat/etaw,xithroat,mrstthroat,xsl0 !uses these, does not set
      common/input/rs,c1,gam
*
      gmm = gam-1.0
      gmp = gam+1.0
      mrstmax = sqrt(gmp/gmm)
*
      if (etaw .eq. 0.0) stop 'GETSTR: etaw=0, fatal'
      eden = 2.*rs*c1
*
```

```
*     Calculate hr and its derivatives and mr and its derivatives.  Also
*     calculate the value of xi corresponding to the given value for
*     mrstar from the do loop.
*
*     write (*,*) 'GETXY: received mrst,eta= ',mrst,eta
      if (mrst .ge. mrstmax) then
        write (*,*) 'GETXY: mrst,mrstmax= ',mrst,mrstmax
        stop 'GETXY: mrst ge mrstmax'
      end if
      if (mrst .le. 0.0) stop 'GETXY: mrst le zero'
      hr = SQRT((1./mrst)*(0.5*gmp - 0.5*gmm*mrst**2)**(-1./(gmm)))
      argxi1 = c1/(c1 - hr + 1.)
      if (argxi1 .ge. 0.0) then
        argxi = 2.*rs*c1*LOG(argxi1)
      else  !singular
        write (*,*) 'GETXY: argxi1= ',argxi1,' mrst= ',mrst
        write (*,*) 'and hr= ',hr,' rs,c1= ',rs,c1
        stop 'this causes fatal error, argxi1 (of log) lt 0.0'
      end if
      if (argxi .ge. 0.0) then
        xi = SQRT(argxi)
      else
        write (*,*) 'GETXY: argxi= ',argxi,' mrst= ',mrst
        write (*,*) 'and hr= ',hr,' rs,c1= ',rs,c1
        stop 'this causes fatal error, argxi lt 0.0'
      end if
      if(mrst .LE. 1.) then
        xi = -xi
      endif
      eterm = EXP(-xi**2/eden)
      hrpr = (xi/rs)*eterm
      hrdpr = (1./rs)*eterm - (xi**2/(rs**2*c1))*eterm
      hrtpr = -(3.*xi/(rs**2*c1))*eterm + (xi**3./(rs**3*c1**2))*
     +       eterm
      hrfpr = -(3./(rs**2*c1))*eterm + (6.*xi**2/(rs**3*c1**2))*
     +      eterm - (xi**4/(rs**4*c1**3))*eterm
      mr = SQRT(2*mrst**2/(gmp - gmm*mrst**2))
      ddc1 = 0.5*gmp - 0.5*gmm*mrst**2
      dmrdmrst = (2.*mrst*gmp/(mr*(gmp - gmm*mrst**2)**2))
      if(mrst .LT. 0.99 .OR. mrst .GT. 1.01) then
* use special formula derived near singularity in derivatives
        dmrstdhr = 2./(mrst*hr/ddc1 - hr/mrst)
        mrpr = dmrdmrst*dmrstdhr*hrpr
        mrstpr = mrpr/dmrdmrst
        ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +          2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
        mdc1 = gmp*mr
        mdc1pr = gmp*mrpr
        mdc2= mrst**2
        mdc2pr = 2.*mrst*mrstpr
        mdc3 = hr
        mdc3pr = hrpr
        mdc4 = hr/mr**2
        mdc4pr = (hrpr*mr - 2.*hr*mrpr)/(mr**3)
        mrdpr = ((mdc1pr*(mdc2*(mdc3-mdc4))-mdc1*(mdc2pr*(mdc3-mdc4)+
     +        mdc2*(mdc3pr-mdc4pr)))/((mdc2*(mdc3-mdc4))**2))*hrpr+
     +        dmrdmrst*dmrstdhr*hrdpr
      else
        mrpr = dmrdmrst*SQRT(2./(gmp*rs))
        mrstpr = mrpr/dmrdmrst
```

```
      ddmrdmrst = (4.*mrstpr*gmp*ddc1 - 4.*mrst*gmp*(mrpr*ddc1 -
     +           2.*mr*gmm*mrst*mrstpr))/(mr**2*(2.*ddc1)**3)
      mrdpr = ddmrdmrst*SQRT(2./(gmp*rs))
     endif
*
*    Calculate the values for theta1, theta3, theta3pr, y1, y3, x2,
*    x4, q2, q4 from Eqn #17a-#17d.  All of these remain constant at
*    any eta for a given value of xi since they are only dependant
*    on hr, mr, and their derivatives.
*
     thc1 = hrdpr
     thc1pr = hrtpr
     thc2 = hr*hrpr*mr**2
     thc2pr = (hrpr*mr)**2 + hr*hrdpr*mr**2 + 2*hr*hrpr*mr*mrpr
     thc3 = hr**2*mr*mrpr
     thc3pr = 2*hr*hrpr*mr*mrpr + (hr*mrpr)**2 + hr**2*mr*mrdpr
     thc4 = hr**2*hrtpr
     thc4pr = 2*hr*hrpr*hrtpr + hr**2*hrfpr
     thc5 = mr**2-1
     thc5pr = 2*mr*mrpr
     thc6pr = hrpr**2*hrdpr
     theta3pr = 0.25*(thc1pr*(thc2+thc3) + thc1*(thc2pr+thc3pr)) +
     +           0.125*(thc4pr*thc5 + thc5pr*thc4 + thc6pr)
     theta1 = hrpr
     theta3 = (1./4.)*(hrdpr*(hr*hrpr*mr**2 + hr**2*mr*mrpr)) +
     +         (1./8.)*hr**2*hrtpr*(mr**2-1.) + (1./24.)*hrpr**3
     x2 = 0.5*hr*hrpr
     x4 = (3./32)*(hr**2*hrpr*hrdpr*(mr**2-1)) - (1./96.)*(hr*
     +     hrpr**3) + theta3*hr*0.25
     y1 = hr
     y3 = (hr/8)*(hr*hrdpr*(mr**2-1) - hrpr**2)
     q2 = 0.5*hr*hrdpr
     q4 = 0.25*(hr**2*hrdpr**2) + (3./32.)*(hr**2*hrdpr**2*(mr**2-
     +     1)) + (1/32.)*(hr*hrdpr*hrpr**2) + 0.25*hr*theta3pr
     x2pr = (0.5*(hrpr**2 + hr*hrdpr))
     x4c1 = (3./32.)*((mr**2-1)*(2.*hr*hrpr**2*hrdpr +hr**2*
     +       hrdpr**2+hr**2*hrpr*hrtpr)+2.*hr**2*hrpr*hrdpr*mr*mrpr)
     x4c2 = (1./96.)*(hrpr**4 + 3.*hr*hrpr**2*hrdpr)
     x4c3 = 0.25*(theta3pr*hr + theta3*hrpr)
     x4pr = (x4c1 - x4c2 + x4c3)
     y1pr = hrpr
     y3pr = (hrpr/8.)*(hr*hrdpr*(mr**2-1) - hrpr**2) + (hr/8.)*
     +       (hrpr*hrdpr*(mr**2-1) + hr*hrtpr*(mr**2-1) + 2*hr*
     +       hrdpr*mr*mrpr - 2*hrpr*hrdpr)
*
*    Calculate Eqn #17.
*
*    Calculate Eqn #17a.  Then, if mrstar is less than one, make the value
*    for x the negative value for plotting purposes.  This is because
*    the value for xi is always positive along the axis.  However, since
*    the paper makes the assumption that the sonic line on the axis occurs
*    at x = 0 (pp. 1339 paragraph before section #3, and figure #3), it
*    is known that for all values of mrstar less than 1, the x value
*    should be negative.
*
     x = xi - x2*eta**2 - x4*eta**4
*
*    Shift the x axis by xsl0 so that the point where x = 0 corresponds
*    to the nozzle throat.
*
```

```
      x = x - xsl0
*
*     Calculate Eqn #17b.
*
      y = y1*eta + y3*eta**3
*
*     Calculate Eqn #17c.
*
      theta = ATAN((y1pr*eta+y3pr*eta**3)/(1.-x2pr*eta**2 -
     +                  x4pr*eta**4))
*
*     Calculate Eqn #17d.
*
* this gives the local mach number, which depends on eta as well
*  as on xi.
      mst = mrst*(1 + q2*eta**2 + q4*eta**4)
*
      return
      end
```